Volume 14 Number 2

ACTA CYBERNETICA

Szeged, 1999

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Contributions are accepted for review with the understanding that the same work has not been published elsewhere.

Manuscripts must be in English and should be sent in triplicate to any of the Editors. On the first page, the *title* of the paper, the *name(s)* and *affiliation(s)*, together with the *mailing* and *electronic address(es)* of the author(s) must appear. An *abstract* summarizing the results of the paper is also required. References should be listed in alphabetical order at the end of the paper in the form which can be seen in any article already published in the journal. Manuscripts are expected to be made with a great care. If typewritten, they should be typed double-spaced on one side of each sheet. Authors are encouraged to use any available dialect of TEX.

After acceptance, the authors will be asked to send the manuscript's source TEX file, if any, on a diskette to the Managing Editor. Having the TEX file of the paper can speed up the process of the publication considerably. Authors of accepted contributions may be asked to send the original drawings or computer outputs of figures appearing in the paper. In order to make a photographic reproduction possible, drawings of such figures should be on separate sheets, in India ink, and carefully lettered.

There are no page charges. Fifty reprints are supplied for each article published.

Publication information. Acta Cybernetica (ISSN 0324–721X) is published by the Department of Informatics of the József Attila University, Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. For 1999 Numbers 1–2 of Volume 14 are scheduled. Subscription prices are available upon request from the publisher. Issues are sent normally by surface mail except to overseas countries where air delivery is ensured. Claims for missing issues are accepted within six months of our publication date. Please address all requests for subscription information to: Department of Informatics, József Attila University, H-6701 Szeged, P.O.Box 652, Hungary. Tel.: (36)-(62)-420-184, Fax:(36)-(62)-420-292.

URL access. All these information and the contents of the last some issues are available in the Acta Cybernetica home page at http://www.inf.u-szeged.hu/local/acta.

H. Bunke
Universität Bern
Institut für Informatik und
angewandte Mathematik
Längass strasse 51., CH-3012 Bern
Switzerland

B. Courcelle
Universitè Bordeaux-1
LaBRI, 351 Cours de la Libèration
33405 TALENCE Cedex
France

J. Demetrovics
MTA SZTAKI
Budapest, P.O.Box 63
H-1502 Hungary

B. Dömölki
IQSOFT
Budapest, Teleki Blanka u. 15-17.
H-1142 Hungary

J. Engelfriet
Leiden University
Computer Science Department
P.O. Box 9512, 2300 RA Leiden
The Netherlands

Z. Ésik
A. József University
Department of Foundations of
Computer Science
Szeged, Aradi vértanúk tere 1.
H-6720 Hungary

L. Lovász
Eötvös Loránd University
Budapest Múzeum krt. 6-8.
H-1088 Hungary

G. Păun
Institute of Mathematics
Romanian Academy
P.O.Box 1-764, RO-70700
Bucuresti, Romania

A. Prékopa
Eötvös Loránd University
Budapest, Múzeum krt. 6-8.
H-1088 Hungary

A. Salomaa
University of Turku
Department of Mathematics
SF-20500 Turku 50, Finland

L. Varga
Eötvös Loránd University
Budapest, Múzeum krt. 6-8.
H-1088 Hungary

H. Vogler
Dresden University of Technology
Department of Computer Science
Foundations of Programming
D-01062 Dresden, Germany

G. Wöginger
Technische Universität Graz
Institut für Mathematik (501B)
Steyrergasse 30
A-8010 Graz, Österreich

Preface

The first Conference for PhD Students in Computer Science (CSCS) was organized by the Department of Informatics of the József Attila University and held in Szeged, Hungary from July 18 to 22, 1998. The aim of the meeting was to provide a forum for PhD students for presenting their results which they achieved during their studies.

The Program Committee was actually the Doctoral Committee of the Department: Miklós Bartha, Tibor Csendes, János Csirik, József Dombi, Zoltán Ésik, Zoltán Fülöp, Ferenc Gécseg, Balázs Imreh, Attila Kuba, Eörs Máté, and György Turán. The Organizing Committee consisted of László Bernátsky, Tibor Csendes, Csongor Halmai, László Martonossy, Lajos Schrettner, and Mariann Sebő.

There were about 80 participants and 70 talks from 8 countries in many fields of computer science and its applications. Beyond the Hungarian PhD Schools in Informatics, mainly those foreign universities were represented (University of Turku and University of Nis) which have a strong cooperation with the József Attila University. The talks were going in two parallel sections in artificial intelligence, automata and formal languages, computer networks, database theory, discrete mathematics, fuzzy decision support systems, information systems, optimization, picture processing, and software engineering. The talks of the students were completed by 4 plenary talks of leading experts.

Those students who gave a talk had the possibility to submit the paper version of their presentaion for publication in a special issue of *Acta Cybernetica*. Altogether 37 manuscripts were submitted, out of which 18 have been accepted, 18 were rejected and one is still under review. This special issue contains 14 of the accepted papers, while 2 papers already appeared in Volume 13 Number 4 (1998), and 2 further will be published in a later regular issue of Acta Cybernetica.

The full program of the conference, the preliminary proceedings and further information can be found on the web page `http://www.inf.u-szeged.hu/~cscs`.

On the basis of our collected positive experiences, the conference will be organized regularly under similar conditions hopefully with more foreign participation. The next meeting will be held also in Szeged, Hungary, in July 2000.

Tibor Csendes
Chairman of CSCS

Zoltán Fülöp
Managing Editor

Acta Cybernetica 14 (1999) 217-227.

On the Partitioning Algorithm

Béla Csaba *

Abstract

We consider the deterministic and the randomized paging problem. We show the close connection between the partitioning algorithm of McGeoch and Sleator and the *OPT* graph of the problem via a natural framework. This allows us to prove some important properties of the "deterministic" partitioning algorithm. As a consequence of these we prove, that it is a k-competitive deterministic on-line algorithm. Besides, we show an application of the *OPT* graph for a special case of the k-server problem.

1 Introduction

The *paging problem* is defined as follows. We have a two-level memory system with k pages of fast memory, and $n-k$ pages of slow memory. Repeatedly a *request* to a page appears. This request should be satisfied by moving the page to fast memory, if it is in slow memory, i.e., a page fault occurs. In this case a page has to be evicted from fast memory to make room for the new, recently requested one. The paging problem is to decide which page is to be evicted. The cost of a *request sequence* is the number of page faults. Of course, this number depends on the strategy used when deciding which page to evict.

There is a simple optimum paging algorithm, Belady's *MIN* algorithm (see [B]), if one knows the whole request sequence in advance, in the *off-line* case. It is more practical to consider the *on-line* paging problem, when the algorithm has to decide immediately after a page request, without knowing what the future requests will be.

Paging is a special case of the so called k-server problem. In this problem we are given a *finite metric space*: a set V on n points, and a distance function $d(x,y)$ on V^2 satisfying the usual requirements of distances (non-negativity, simmetry, triangle-inequality). There are k $(1 < k < n)$ mobile servers, initially residing on some points of V, no two on the same point. A number of requests, each is an element of V, appears. A request has to be satisfied immediately by moving a server to the requested point, if there is no server on it. By moving a server from the point y to the requested point x a cost, $d(x,y)$ incurs. The total cost of a

*Department of Computer Science; Rutgers, The State University of NJ; 110 Frelinghuysen Road; Piscataway, NJ 08854-8019 USA. Email: bcsaba@paul.rutgers.edu

request sequence is the sum of the costs of the composing requests. One can see, that by choosing $d \equiv 1$ (*uniform metric space*), we arrive to paging. The number of faults done by some algorithm on a request sequence σ is the same as the total distance taken by the servers during the satisfaction of σ. For the case of simplicity we use the terminology of the uniform k-server problem throughout the paper. An important remark is that it is enough to consider *lazy* algorithms, i.e., algorithms, which move a server only to a requested point (see [MMS]).

For comparing two paging algorithms the *competitive ratio* is used. This measure of performance of an on-line algorithm was introduced by Sleator and Tarjan (see [ST]). Fix any starting configuration of the pages, and denote by $opt(\sigma)$ the optimal cost of the satisfaction of the request sequence σ. The competitive ratio of the on-line algorithm $\mathcal{A}$ is c, if there is a constant M such that on every request sequence σ the cost incurred by $\mathcal{A}$, $\mathcal{A}(\sigma)$ is at most $c \cdot opt(\sigma) + M$. It was shown (see [ST]) that no on-line algorithm can have a competitive ratio less than k for the paging problem. $LRU, FIFO$ and a large number of other on-line algorithms are known to be k-competitive. On the other hand, the best competitive ratio achieved by some on-line algorithm for the k-server problem is $2k-1$ (see [KP]), while the lower bound for any metric space is k (see [MMS]), like in paging.

As it happens frequently, one may expect a better performance in the randomized case. A randomized on-line algorithm $\mathcal{R}$ is c-competitive, if there is a constant M such that on every request sequence σ, $E[\mathcal{R}(\sigma)]$ is at most $c \cdot opt(\sigma) + M$, where $E[\mathcal{R}(\sigma)]$ denotes the expected cost incurred by $\mathcal{R}$ on σ. It was proved (see [FKLMSY]) that $H_k = 3D1 + \frac{1}{2} + \cdots + \frac{1}{k}$ is a lower bound for the randomized competitiveness of an on-line paging algorithm. There is a simple, elegant algorithm, which has randomized competitive ratio $2H_k$ (see [FKLMSY]). On the other hand, the only known optimal randomized algorithm, the *partitioning algorithm* has a much more complicated description. Our goal is to show that despite it's complexity, the "deterministic" version of this algorithm is based on rather plausible ideas. This is done by the help of OPT graphs.

OPT graphs for on-line problems were first considered in [CL1], [CL2] when investigating the deterministic k-server problem, and in [LR] were used to analyze the randomized case. Roughly speaking, the OPT graph for the k-server problem is a finite directed graph, with which one can easily compute the optimal cost of any request sequence, and find the corresponding satisfaction.

The outline of the paper is as follows. In the second section we give the definition of the partitioning algorithm, and our framework, which is based on the OPT graph of the problem. The third section deals with the deterministic paging problem, and the connection between the OPT graph and the partitioning algorithm, and we show that the partitioning algorithm is k-competitive in the deterministic case. In the fourth section we give an application of our results for a special k-server problem, achieving $2k-2$ competitive ratio.

2 Definitions

First we give the definition of the partitioning algorithm following [MS]. The algorithm dynamically maintains a partition of the points: $V = S_\alpha \bigcup S_{\alpha+1} \bigcup \ldots \bigcup S_\beta$. Each set S_i $(i < \beta)$ is labeled with an integer k_i. Initially $\alpha = 1$ and $\beta = 2$,S_2 contains the k points that are covered by the servers, S_1 contains the unoccupied points, and $k_1 = 0$. In response to a request at a point $r \in V$ the labeled partition is updated. Let $r \in S_i$, then there are three cases.

Rule 1: $i = \beta$
Do nothing.

Rule 2: $\alpha < i < \beta$

First do the following assignements:

- $S_i \leftarrow S_i - r$
- $S_\beta \leftarrow S_\beta + r$
- $k_j \leftarrow k_j - 1$, $i \leq j < \beta$.

If some label changed from 1 to 0, find the largest number j such that $k_j = 0$. Then let

- $S_j \leftarrow S_\alpha \bigcup S_{\alpha+1} \bigcup \ldots \bigcup S_j$
- $\alpha \leftarrow j$.

Rule 3: $i = \alpha$

Do the following assignements:

- $S_\alpha \leftarrow S_\alpha - r$
- $S_{\beta+1} \leftarrow r$
- $k_\beta \leftarrow k - 1$
- $\beta \leftarrow \beta + 1$.

By induction one can easily show that the following *labeling invariant conditions* always hold:

- $k_\alpha = 0$
- $k_i > 0$ $(\alpha < i < \beta)$
- $k_i = k_{i-1} + |S_i| - 1$ $(\alpha < i < \beta)$
- $k_{\beta-1} = k - |S_\beta|$.

Now we are ready to give the partitioning algorithm itself. Denote $S_\alpha \bigcup S_{\alpha+1} \bigcup \ldots \bigcup S_i$ by S_i^*. For each set S_i^*, where $\alpha < i < \beta$, there are k_i *i-marks* occupying different points of S_i^*. An i-mark is only allowed on the points of S_i or on points with an $(i-1)$-mark - on the *i-eligible* points. The algorithm keeps a server on each element of S_β and on each point with a $(\beta-1)$-mark. By the labeling invariant conditions this means exactly k points covered by some server. Before the first request $\alpha = 1$ and $\beta = 2$, and there are no marks of any kind. Now, let the new request be r.

(1) If $r \in S_\beta$, then do nothing.

(2) If $r \in S_i\,(\alpha < i < \beta)$, move the marks around so as to achieve that there is a j-mark on r for all $j\,(i \leq j < \beta)$ in the following way. If r has a j-mark, then do nothing. Otherwise randomly choose some point w that has a j-mark. Transfer each l-mark $(l \geq j)$ from w to r. Repeat this step, if r does not have all marks up to $\beta - 1$. Then apply **Rule 2**, and erase all the marks on r. If α changes, all t-marks $(t \leq \alpha)$ are erased. If a $(\beta-1)$-mark moves to r from another point, the corresponding server moves to r.

(3) If $r \in S_\alpha$, then apply **Rule 3**. Then create $k-1$ new $(\beta-1)$-marks, and distribute them randomly amongst the k $(\beta-1)$-eligible points. We move the server to r from the point which is left without a $(\beta-1)$-mark.

Recall, that this randomized algorithm is H_k-competitive. Note, that with any deterministic rule for distributing the marks this algorithm switches to a deterministic one. Now we turn our attention to the OPT graph of the paging problem. We repeat the definition of [LR].

Definition 2.1 *An OPT graph of an on-line k-server problem is a finite directed graph with one distinguished vertex I, such that (1) each edge is labeled with a request from V and a cost, (2) for each vertex and each request r there is a unique edge out of that vertex whose request label is r, and (3) for every request sequence ϱ, opt(ϱ) equals the sum of the cost labels on the path given by ϱ starting from I.*

Every legal configuration of the servers can be viewed as a k element subset of the points. If $S \subset V$ is the set of points occupied by the servers, and r is the new request, the k-sets reachable by some algorithm from S by r are the elements of the following set system: $\mathcal{H} = \{H : H = S - s + r,\ s \in S\}$. If $r \in S$, then $S = \mathcal{H}$, otherwise $\mathcal{H}$ has k elements. Denote $\mathcal{H}_0$ the initial configuration of the servers, and let $\varrho = \varrho_1\varrho_2 \ldots \varrho_m$ be a request sequence of lenth m. If $\mathcal{H}_i = \{H : H = H' - -h' + \varrho_i,\ h' \in H' \in \mathcal{H}_{i-1}\}$, where $1 \leq i \leq m$, then every ϱ-satisfying algorithm's movements from configuration to configuration is embedded in the following sequence: $\mathcal{H}_0 \to \mathcal{H}_1 = 20 \to \ldots \to \mathcal{H}_m$. An important observation, that it is possible that in a set system $\mathcal{H}_i$ there is a k-set H for which the cost of any satisfying algorithm starting from $\mathcal{H}_0$ and getting to H is "too big" comparing to that of some other $H' \in \mathcal{H}_i$. One may think that this kind of sets can be ignored without eliminating any actions of a "good algorithm". In what follows these simple ideas will be made precise.

First, we give three rules for building a finite directed graph G. The vertices of

this graph are associated with set systems of k-sets, one distinguished vertex is I, the initial configuration. Now let $\mathcal{H}$ be a vertex of the graph, $r \in V$.

Rule A: If $r \in \bigcap_{H\in\mathcal{H}} H$, then do nothing.

Rule B: If $r \notin \bigcap_{H\in\mathcal{H}} H$, but $r \in \bigcup_{H\in\mathcal{H}} H$, then let $\mathcal{H}' = \{H' : H' \in \mathcal{H},\ r \in H'\}$. If $\mathcal{H}'$ is not present in G, then put it in as a new vertex. Draw the directed edge $(\mathcal{H}, \mathcal{H}')$ in G with label r. We call this kind of edge a decreasing edge.

Rule C: If $r \notin \bigcup_{H\in\mathcal{H}} H$, then let $\mathcal{H}' = \{H' : H' = H - h + r,\ h \in H \in \mathcal{H}\}$. If $\mathcal{H}'$ is not present in G, then put it in as a new vertex. Draw the directed edge $(\mathcal{H}, \mathcal{H}')$ in G with label r. We call this kind of edge an increasing edge.

Note, that **Rule B** stands for discarding the "expensive k-sets", and we apply **Rule C**, when we are forced to move a server. Starting from I after a finite number of applications of the rules we arrive to a graph G to which we can't put new vertices or edges. As one can see, there cannot be more than $n \cdot 2^{\binom{n}{k-1}}$ vertices of G.

3 Properties of OPT graphs and an on-line algorithm

Call a vertex of the graph G defined in the previous section a *single vertex*, if it contains only one configuration, otherwise call it a *multiple vertex*. The following two lemmas prove, that G is the OPT graph of the problem.

Lemma 3.1 *Let ϱ be a request sequence, and assume that starting from I we arrive to the vertex $\mathcal{H}$ following ϱ in G, and $H \in \mathcal{H}$ is a configuration. Then there exists a satisfaction of ϱ with endconfiguration H and its cost is the number of increasing edges in the above walk.*

Proof. Let's go backwards from H on the walk determined by ϱ. From the definition of decreasing edges it follows that until we reach an increasing edge, we don't have to change configuration. When an increasing edge is coming, it is enough to move a server, and we can get to the configuration, from which this increasing edge is going out. Hence, for the decreasing edges on the walk there is no incurring cost, and in the case of increasing edges the cost is 1. □

Lemma 3.2 *Let ϱ be a request sequence, and assume that starting from I we arrive to the vertex $\mathcal{H}$ in G. Then the set system $\mathcal{H}$ contains exactly the optimally reachable configurations, and the optimal cost is the number of increasing edges traversed when getting to $\mathcal{H}$.*

Proof. We proceed by induction on the length of the request sequence. If $|\varrho| \leq 1$, then the lemma trivially holds. Let's suppose that it is true for every request sequence with length at most t, and $|\varrho| = t$.

First we prove, that for every new request r, if the edge $(\mathcal{H}, \mathcal{H}')$ is labeled r, then for $H' \in \mathcal{H}'$ the optimal cost of arriving to H' is the number of increasing edges.

Let's assume that there is a satisfaction of ϱr which arrives to H' with cost less than the number of increasing edges. Denote the jth configuration of this satisfaction by Q_j. If $Q_t \notin \mathcal{H}$, then by the induction hypothesis we know that the cost of getting to Q_t by ϱ is bigger than the number of increasing edges, because every optimally reachable configuration is the element of $\mathcal{H}$. Hence, after a new request the cost of getting to $Q_{t+1} = H'$ is at least as big as the number of increasing edges. If $Q_t \in \mathcal{H}$, then the statement trivially holds, either the edge $(\mathcal{H}, \mathcal{H}')$ is an increasing or a decreasing edge.

So far we have proved that the optimal cost for $\mathcal{H}'$ is the number of increasing edges. Let's assume that there is a satisfaction $\mathcal{S}$ of ϱr such that S_{t+1}, the last configuration of it is not in $\mathcal{H}'$, but it has optimal cost. If the cost of $\mathcal{S}$ were smaller, than the number of increasing edges, then because up to S_t the cost can't be smaller than the cost of $\mathcal{H}$ (by the induction hypothesis), $S_t \in \mathcal{H}$. One can easily see, that r has to be an increasing request, but then the cost of $\mathcal{S}$ cannot be smaller than the cost of $\mathcal{H}'$. Hence, the only case is when $S_{t+1} \notin \mathcal{H}'$, and has the same cost. Denote the directed walk from I by $\mathcal{H}_0(= I) \to \mathcal{H}_1 \to \ldots \to \mathcal{H}_t(= \mathcal{H}) \to \mathcal{H}_{t+1}(= \mathcal{H}')$. There is a last configuration of $\mathcal{S}$ which is the element of the corresponding vertex of the graph. Denote it by S_m, and let $r_1, r_2, \ldots r_l$ be the last $l = t + 1 - m$ requests. Thus, $S_m \in \mathcal{H}_m$, but for $j \geq 1$ $S_{m+j} \notin \mathcal{H}_{m+j}$. By our assumption the cost incurred on $\mathcal{S}$ before the last request is greater than that incurred on the ϱ-path in G. Hence, the last edge, $(\mathcal{H}_t, \mathcal{H}_{t+1})$ is an increasing edge, and $S_t = S_{t+1}$. Denote R_i the closest set in $\mathcal{H}_i$ to S_i, i.e., $|R_i \bigcap S_i| = max\{|H \bigcap S_i| : H \in \mathcal{H}_i\}$. Observe, that $r_1 \notin S_m$, and the $(\mathcal{H}_m, \mathcal{H}_{m+1})$ edge is a decreasing edge, otherwise $S_1 \in \mathcal{H}_{m+1}$. From this follows, that $|R_1 \bigcap S_1| = k - 1$. There are five cases to consider when satisfying the last l requests. In the following $1 \leq i \leq l - 1$.

(1) $r_{i+1} \in S_i \bigcap R_i \Longrightarrow |S_i \bigcap R_i| = |S_{i+1} \bigcap R_{i+1}|$, and the difference of the costs doesn't change (lazy satisfactions).

(2) $(\mathcal{H}_i, \mathcal{H}_{i+1})$ is an increasing edge, and $r_{i+1} \in S_i \Longrightarrow$ the difference of the costs is decreased by 1, and $|S_i \bigcap R_i| + 1 = |S_{i+1} \bigcap R_{i+1}|$. This equality easily follows from the definition of G (**Rule C**).

(3) $(\mathcal{H}_i, \mathcal{H}_{i+1})$ is an increasing edge, and $r_{i+1} \notin S_i \Longrightarrow$ the difference of the costs doesn't change, and $|S_i \bigcap R_i| \leq |S_{i+1} \bigcap R_{i+1}|$.

(4) $r_{i+1} \notin S_i$, and $r_{i+1} \in R_i \Longrightarrow$ the difference of the costs is increased by 1, and $|S_i \bigcap R_i| \leq |S_{i+1} \bigcap R_{i+1}|$.

(5)$r_{i+1} \notin S_i$, $(\mathcal{H}_i, \mathcal{H}_{i+1})$ is a decreasing edge, but $R_i \notin \mathcal{H}_{)+\infty} \Longrightarrow$ the difference of the costs is increased by 1, and $|S_i \bigcap R_i| \leq |S_{i+1} \bigcap R_{i+1}| + 1$.

Let us suppose first, that at the jth stage $(1 \leq j \leq l)$ the difference of the costs is $\geq k$. Then by moving at most $k - 1$ servers from R_j we can reach S_j (the intersection $S_i \bigcap R_i$ always contains the most recently requested point), and this has cost at most $k - 1$. Hence, there is a ϱ-satisfying configuration sequence to S_{t+1} with smaller cost, and this contradicts with the optimality of $\mathcal{S}$. Let us consider now the quantity $D_i = k - |S_i \bigcap R_i|$. We claim, that D_i is a lower bound for the difference of the costs in the ith stage. For $i = 1$ this is obviously true. Assume, that D_i is a lower bound up to the ith step, and a new request, r_{i+1} is

coming. We check the possible five cases. In case (1) $D_i(= D_{i+1})$ certainly remains a lower bound. In case (2) the drawback of $\mathcal{S}$ is decreased, but the intersection size increases by one. In case (3) the drawback doesn't change, and the intersection size may increase. In cases (4)-(5) the drawback increases, but the size of the intersection doesn't decrease by more than one. Thus, in all cases D_{i+1} is a lower bound. By our assumptions, $D_{l-1} \leq 1$, and the last request should be a request considered in case (2). By the definition of G, $S_{t+1} \in \mathcal{H}_{t+1}$. We get, that $\mathcal{H}_{t+1}$ consists of exactly the optimally reachable configurations. □

Definition 3.3 *Consider the partition $S_\alpha \bigcup S_{\alpha+1} \bigcup \ldots \bigcup S_\beta$ given by the partitioning algorithm. Let $\mathcal{P} = \{P : |P| = k - |S_\beta|$, P does not contain more than k_i points from $S_i^*\}$. The set system given by the partition is $\mathcal{S} = \{S : S = S_\beta \bigcup P, \in \mathcal{P}\}$.*

Lemma 3.4 *Let $\mathcal{S}$ be the set system determined by the partitioning algorithm. Then $S_\beta = \bigcap_{S \in \mathcal{S}} S$.*

Proof. $S_\beta \subseteq \bigcap_{S \in \mathcal{S}} S$ trivially holds. Let's suppose that for some $v \in V$ $v \in \bigcap_{S \in \mathcal{S}} S$, and $v \in S_i$, $\alpha < i < \beta$. By the labeling invariant conditions $k_i = k_{i-1} + |S_i| - 1$, i.e., $|S_i| = k_i - k_{i-1} + 1$. If $v \in \bigcap_{S \in \mathcal{S}} S$, then every point of S_i is in the intersection, because they have the same role. Thus, from S_{i-1}^* we could choose only $k_{i-1} - 1$ points. On the other hand we are allowed to choose k_{i-1} points — we arrived at a contradiction. □

Lemma 3.5 *Assume, that starting from I we follow the edges of G determined by the request sequence ϱ, and we arrive to the vertex $\mathcal{H}$. Let the set system given by the actual partition (after ϱ) is $\mathcal{S}$. Then $\mathcal{H} = \mathcal{S}$.*

Proof. We prove the statement by comparing the maintaining rules for the partitioning algorithm and the build-up rules for G, by induction on the length of ϱ. If $|\varrho| = 0$, the lemma trivially holds. Let's suppose now that the lemma is true for ϱ, a new request r is coming, the edge $(\mathcal{H}, \mathcal{F})$ is labeled r in G, and the new set system given by the partition is $\mathcal{Q}$. We distinct three cases depending on the rule we use to maintain the partition.

If $r \in S_\beta$: There is nothing to prove, $\mathcal{F} = \mathcal{H}$ and $\mathcal{Q} = \mathcal{S}$ (**Rule 1** $\Leftrightarrow$ **Rule A**).

If $r \in S_i\,(\alpha < i < \beta)$: By **Rule 2**, $Q_\beta = S_\beta + r$, and $Q_i = S_i - r$ and $k_j \leftarrow k_j - 1$ for $j : i \leq j < \beta$. If for some j, k_j has become 0, this means, that previously k_j was 1, and hence, from S_j^* we could choose only one point to some $S \in \mathcal{S}$. By $Q_\alpha = S_\alpha \bigcup S_{j_0}^*$, where j_0 is the biggest such j, we discard all the sets $S \in \mathcal{S}$ which doesn't contain r. Thus, the set system Q contains exactly those sets $S \in \mathcal{S}$, for which $r \in S$. Using the induction hypothesis and the definition of **Rule B**, we get, that $\mathcal{Q} = \mathcal{F}$.

The only possibility left is that $r \in S_\alpha$. By **Rule 3**, $Q_\beta = r$, thus, $r \in Q$ for every $Q \in \mathcal{Q}$, and this is the only element of the intersection of the sets of $\mathcal{Q}$. Also, $k_\beta \leftarrow k_\beta - 1$ and $\beta \leftarrow \beta + 1$, hence $\mathcal{Q}$ contains exactly the sets which has r and other $k - 1$ points from some $S \in \mathcal{S}$. But this is the set system $\mathcal{F}$ we get from $\mathcal{H}$ by applying **Rule C**. □

Definition 3.6 *Let H and F be two configurations. We say, that F is achievable from H (and vice versa), if $|H \bigcap F| = k-1$, i.e., moving one server is enough to reach one from the other.*

Lemma 3.7 *Let $\mathcal{H}$ be a vertex in G, $r \in V$, and $H \in \mathcal{H}$. If $(\mathcal{H}, \mathcal{F})$ is the outgoing edge from $\mathcal{H}$ labeled r, then there is an achievable $F \in \mathcal{F}$ from H.*

Proof. If $\mathcal{H} = \mathcal{F}$ (**Rule A**), then there is nothing to prove
If $(\mathcal{H}, \mathcal{F})$ is an increasing edge, then there are k sets in $\mathcal{F}$ which are achievable from H: all the sets of the form $F = H - h + r$, $h \in H$.

Let's suppose now, that $(\mathcal{H}, \mathcal{F})$ is a decreasing edge. If $r \in H$, then $H \in \mathcal{F}$, so, let $r \notin H$. By Lemma 3.5 (and using the notation of it), $r \in S_i$ for some $i : \alpha < i < \beta$.

There are two possibilities.

(1) When composing H, we did not pick any points from S_i^*. But then there is a largest t ($t \geq 0$), such that we did not choose any point from S_{i+t}^*, because we had the necessary number of points. Hence, we picked a point $v \in S_{i+t+1}^*$. Substituting this v by r ($r \in S_{i+t+1}^*$) we arrive to a set H', for which $|H \bigcap H'| = k-1$, and $r \in H'$. Thus, $H' \in \mathcal{F}$, and it is achievable from H.

(2) We picked another point w from S_i^*. Substituting w by r, we again get a set of $\mathcal{F}$, which is achievable from H. □

Lemma 3.8 *There cannot be more than $k-1$ consecutive decreasing edges in any walk on G.*

Proof. Notice, that for any vertex $\mathcal{H}$ in G, $r \in \bigcap_{H \in \mathcal{H}} H$ if r is the label of an ingoing edge. After each application of **Rule B** this intersection size increases. Thus, after $k-1$ decreasing edges we arrive to a vertex which is represented by a single k-set. From such vertices every outgoing edge is an increasing edge: no k consecutive decreasing edges are possible. □

Now we are ready to discuss our on-line algorithm. Roughly speaking, we do a walk on the OPT graph step by step according to the incoming requests. We introduce some notation: r is the new request, S is the configuration of the servers, and $\mathcal{H}, \mathcal{H}'$ are vertices of the OPT graph.

- (1) Initially $S = \mathcal{H} = I$.
- (2) If $r \in S$, then no server moves. If r is a loop edge label of $\mathcal{H}$ (the actual vertex of the graph), then we stay there. If $(\mathcal{H}, \mathcal{H}')$ is a decreasing edge labeled r, then $\mathcal{H}'$ will be the new actual vertex.
- (3) If $r \notin S$, then $(\mathcal{H}, \mathcal{H}')$ is either a decreasing edge or an increasing edge labeled r. Choose any configuration $H' \in \mathcal{H}'$, which is reachable from S. We know, that there is always at least one such configuration (Lemma 3.7). Move the server on $S - H$ to r.

Note, that in (3) there is no deterministic rule how to choose the server to move, when there is a multiple choice. We will see, that one can use any kind of rule in these cases.

Theorem 3.9 *The algorithm described above is a k-competitive on-line algorithm.*

Proof. From the description it is obvious, that the algorithm is well defined, and in an on-line fashion we never have to move more than one server for a request. We decomp ose the walk defined by the request sequence ϱ in the OPT graph into s everal *phases*. The first phase starts from I, consists of the first consecutive increasing edges, and the conse cutive decreasing edges coming right after them. This phase ends, when a new increasing edge is to be traversed. =20 Then a new phase starts, with the same structure: consecutive increasing edges, and then consecutive decreasing edges. Observe, that from Lemma 3.1 and 3.2 it follows, that the optimal cost of ϱ is the number of increasing edges traversed while satisfying ϱ. In a phase by definition, there is at least one increasing edge, and not more, than $k-1$ decreasing edges. This is the consequence of Lemma 3.8. As mentioned above, the algorithm never moves more than one server for a request. Hence, in every phase the cost of the optimal satisfaction is the number of increasing edges, and the cost of our on-line algorithm is at most the sum of the umber of the increasing and decreasing edges (at most, because the algorithm not necessarily moves a server for a decreasing request). It is easy to see, that the fraction of these two quantities is always at most k. From this the theorem follows. □

Remark: Say, that there are i increasing edges in a phase. Then the compe titive ratio for that phase is at most $\frac{k-1+i}{i} < \frac{k}{i} + 1$. When n is large enough comparing to k, then we expect more than one increasing edge and less than $k-1$ decreasing edges in an "average phase". Hence, this algorithm works well in these cases. Unfortunately, either storing the OPT graph in memory or dynamically computing the next vertex needs a lot of resources. Thus, this algorithm is undesirable in practice, but possibly of theoretical interest. It suggests, that for "random sequences" the competitive ratio of an on-line algorithm can be much smaller, than k.

4 Application for a k-server problem

In this section we use the OPT graph of the paging problem to define an on-line algorithm for a special case of the k-server problem. Let us call a finite metric space *multipartite*, if the points can be distributed into several classes, where the distance between two points is 1, when they correspond to different classes, and any number in the [1,2] interval otherwise. One can easily show, that these are valid metric spaces, that is, non-negativity and symmetry of the distances, and the triangle inequality are satisfied.

Theorem 4.1 *If in a multipartite metric space no class has more than $k-1$ elements, then there is an on-line algorithm for the k-server problem of this metric space with competitive ratio $2k-2$.*

Proof. Our algorithm is almost the same, which was considered in the paging problem, but now we have less freedom in the multiple choices. If the algorithm is forced to move, then we try to move distance 1 if it is possible. Otherwise, we move any server consistently with the actual OPT graph vertex. Observe, that if the new request is an increasing request, then we can choose a server which move s distance 1. There are at most $k-1$ points in one class, hence, at least two servers are not in the class of the new request. Another important case, when a phase starts from a single vertex H, and that phase has only one increasing edge. That edge goes to a vertex $\mathcal{H}'$, which contains exactly the union of the $k-1$ element subsets of H and r, the new request. Whichever server we have moved to r, there is always another server from another class in another configuration of $\mathcal{H}'$. Hence, for the first decreasing edge we either don't move a server at all, or there is a server, which has to move only 1. Again, this is a simple consequence of the class sizes. Thus, in such phases the optimal cost is at least 1, while our on-line cost is at most $1+1+2(k-2)$, from which we have the $2k-2$ bound for the competitive ratio of these phases.

If there are more than one increasing edge in a phase, then the competitive ratio of such a phase is at most $\frac{i+2(k-1)}{i} \leq 1+k-1=k$, where i is the number of increasing edges. When there are less than $k-1$ decreasing edges, the competitive ratio of the phase is at most $\frac{i+2(k-2)}{i} \leq 2k-3$.

There is one case left: phases with one increasing edge and $k-1$ decreasing edges, starting from a multiple vertex. If a phase starts from a multiple vertex, then the previous phase has at most $k-2$ decreasing edges. We compute the competitive ratio of these two consecutive phases. It is at most $\frac{1+2(k-2)+1+2(k-1)}{2} = 2k-2$. □

This result was an illustration, the careful reader may notice that by decreasing the class sizes, a more thorough analysis gives smaller competitive ratios. On the other hand, we cannot expect a k-competitive algorithm for non-uniform metric spaces by just using the OPT graph of the paging problem. OPT graphs for non-uniform spaces may prove to be useful, but up to this time these graphs were investigated only for very special cases. Notice, that for non-uniform problems we may lose the symmetry of the graph, that can make the analysis hard.

Let us discuss a little bit more on the connection of paging and the general k-server problem. In a finite metric space divide every distance with the length of the smallest distance in it. This way every distance will be in the [1,D] interval for some D. Let ϱ be a request sequence, and denote $opt_p(\varrho)$ the optimal cost of ϱ in the uniform metric space, while $opt(\varrho)$ denotes the optimal cost of satisfaction in the original one. Then $opt_p(\varrho) \leq opt(\varrho)$ and $opt(\varrho) \leq D \cdot opt(\varrho)$, obviously. If $\mathcal{A}$ is any k-competitive paging algorithm, then $\mathcal{A}(\varrho) \leq k \cdot opt_p(\varrho) + M$ for some constant M, and thus $\mathcal{A}(\varrho) \leq D \cdot k \cdot opt(\varrho) + M$. Thus, reaching the $2 \cdot k$-competitivity for multipartite metric spaces is easy, any k-competitive paging algorithm achieves it.

5 Summary

In this paper we investigated the paging problem, and a special k-server problem. We used OPT graphs to have a better insight to paging. Our results suggests, that the partitioning algorithm in practice may perform well, considering only the competitivity as a measure. Then we proved a new nontrivial upper bound for the multipartite k-server problem. We did it by the help of the OPT graph of the paging problem. While our opinion is that one cannot expect much more with our technique, we think, that a better understanding of the structure of OPT graphs for non-uniform spaces may result in better upper bounds.

Acknowledgement The author is grateful Péter Hajnal and Endre Szemerédi for listening to earlier versions of this paper, and to Tibor Széles for his valuable help in the proofreading.

References

[B] Belady, L., *A study of page replacement algorithms for virtual storage computers*, IBM Systems Journal, 5:78-101, 1966

[CL1] Chrobak, M., Larmore, L., *The Server Problem and On- -line Games*, Proceedings of the DIMACS Workshop on On-line Algorithms, American Mathematical Society, February 1991

[CL2] Chrobak, M., Larmore, L., *Generosity helps, or an 1 1-competitive algorithm for three servers*, Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms, 1992

[FKLMSY] Fiat, A., Karp, R., Luby, M., McGeoch, L., Sleator, D., Young, N., *Competitive Paging Algorithms*, Journal of Algorithms 12 (1991), pp. 685-699.

[KP] Koutsoupias, E. and Papadimitrou, C., *On the k-Server Conjecture*, STOC 94, pp. 507-511.

[LR] Lund, C., Reingold, N., *Linear Programs for Randomized On-line Algorithms*, Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, 382-391, 1994

[MMS] Manasse, M. S., McGeoch L. A. and Sleator, D. D., *Competitive Algorithms for Server Problems*, Journal of Algorithms 11 (1990), pp. 208-230.

[MS] McGeoch L. A. and Sleator, D. D., *A Strongly Competitive Randomized Paging Algorithm*, Algorithmica (1991), pp. 816-825.

[ST] Sleator, D. D., Tarjan, R. E., *Amortized Efficiency of List Update and Paging Rules*, Comm. of the ACM, February 1985, pp. 202-208.

Acta Cybernetica **14** (1999) 229-238.

Test Suite Reduction in Conformance Testing

Tibor Csöndes * Sarolta Dibuz * Balázs Kotnyek †

Abstract

Conformance testing is based on a test suite. Standardization committees release standard test suites, which consist of hundreds of test cases. The main problem of conformance testing is that we do not have enough time to execute them all. Therefore, test selection is required to maximize the test coverage. In our earlier papers [6,7] we outlined a new method of selecting an optimal test suite which can detect the errors with better probability and reduce the time of testing. In this paper we will expound the mathematical optimization method for test suite optimization based on cost and test coverage, and we will apply this method to an ISDN protocol.

1 Introduction

The main aim of conformance testing is to check whether the protocol implementation conforms to the standard. The procedure of conformance testing as well as the protocols are standardized [1]. The two main terms of testing are the test purpose (TP) and the abstract test suite (ATS). The test purpose is the description of the well defined objective of testing to focus on a single conformance requirement or a set of related conformance requirements. The ATS consists of several test cases (TC) created to test one or more TPs. In real-life conformance testing, the testers choose some of the TPs and execute all the TCs that are related to the chosen set of TPs. The challenge in conformance testing is this selection, choosing this set so that the coverage, the fault detection capability, be maximal. Of course, the best selection is when we choose all the TPs or - what means the same - all the TCs. The problem that arises here is the time limitation. Usually we do not have enough time to do this, so we can only execute some of the TCs.

The existing approach of handling this problem is the test generation. The goal of such procedures is briefly to generate optimal ATSs from the protocol specification , i.e. that contain as few parallelisms as possible so they can be entirely executed within the time limit. The theoretical background of this kind of optimization is the finite state machine (FSM). If a FSM model of the protocol is already given, there are several algorithms for generating good or better test suites (Transition Tour, Unique Input/Output method, Distinguishing Sequence) [4]. We

*Conformance Center, Ericsson Ltd.,Laborc u. 1., H-1037 Budapest, Hungary
†Eötvös Loránd University, Budapest

need this FSM model, however and real life protocols are so complicated that it is not possible to create their usable FSM model.

Our approach, which we outlined in our earlier papers [6,7], is based on practice. We suppose that we are given an ATS and we cannot generate any new TCs. That is what test laboratories do, they use only those test cases which are provided by the standards. By now, the selection of TCs from ATS is based on the subjective decision of the test laboratory. Our aim is to create a theoretical background of such selection. We found that mathematical optimization could be a suitable one.

The rest of the paper is organized as follows: first, we introduce a model of conformance testing mathematics could operate on. In Section 3 we describe this operation, in Section 4 we present how this method could be applied to an existing protocol the ISDN DSS1 Layer 2 protocol[2]. We chose this protocol because it is widely used, well known, and its ATS exists.

2 The model

In our model, we create a mathematically formulated relation between the test cases and test purposes by introducing the purpose-test incidence matrix. The purposes are placed in the rows and the tests in the columns. As a result we get an A matrix of size $k \times n$ where $P_1, \ldots, P_k$ are the purposes and $T_1, \ldots, T_n$ are the test cases defined in the ATS (Figure 1). The j^{th} element in the i^{th} row is 1 if, and only if T_j is necessary to check P_i, otherwise it is 0. In other words, if we want to check a purpose (e.g. P_i) completely, we have to execute all test cases having 1 in the row of the purpose (namely in the i^{th} row). Let us introduce the number b_i designating the number of such test cases and let $b = (b_1, \ldots, b_k)^T$ be the vector made of these numbers.

	T_1	T_2	$\cdots$	T_n
P_1	1	0	$\cdots$	1
P_2	1	1	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
P_k	0	1	$\cdots$	0

$= A$

Figure 1: Purpose-test incidence matrix

There are protocols the ATSs of which define one-to-one connection between the test cases and the purposes, hence their incidence matrix is diagonal. There exist, however, ATSs the matrices of which are not diagonal, thus there are more than one necessary test cases to check a test purpose. In this paper, we are dealing with these kind of protocols.

Let us assign the value $cov(P_i)$, $(i = 1, \ldots, k)$ to every purpose describing its coverage. This value may be obtained from a theoretical consideration or we can simply mark the priority of the purpose with it. Similarly, let us designate the cost

of test T_j with $c(T_j)$, $(j = 1, \ldots, n)$. This cost function can be defined to represent the resources (time) required to execute the test case. The cost of a set of test cases can be defined simply as the sum of the cost of the individual test cases in the set. Let us do the same with the coverage of a set of purposes.

Let us introduce the increasing functions $f_i : \{0, 1, \ldots, b_i\} \to [0, cov(P_i)]$, $(i = 1, \ldots, k)$ describing the coverage we get if we execute m tests among the tests that correspond to P_i $(m = 0, 1, \ldots, b_i)$. Of course $f_i(0) = 0$ and $f_i(b_i) = cov(P_i)$ for all $i = 1, 2, \ldots, k$. The different models differ from each other in choosing functions f_i. We introduce three models below.

1. **Linear model:** The coverage is in direct proportion to the number of executed tests i.e.

$$f_i(m) = \frac{m}{b_i} cov(P_i) \qquad m = 0, \ldots, b_i$$

2. **"All or nothing" model:** We only consider the purpose being checked when we executed all the necessary tests.

$$f_i(0) = f_i(1) = \ldots = f_i(b_i - 1) = 0 \quad \text{and} \quad f_i(b_i) = cov(P_i)$$

3. **"One is enough" model:** If only one test is executed among the ones that correspond to P_i, we get the whole coverage.

$$f_i(0) = 0 \quad \text{and} \quad f_i(1) = \ldots = f_i(b_i) = cov(P_i)$$

3 Optimization

We introduce two possible optimization problems. In the first one our aim is to select a test set from the test suite with **minimal cost** supposing a constraint bounding the coverage from below. Let $x \in \{0, 1\}^n$ be the decision vector, so $x_j = 1$ if T_j is executed and $x = 0$ if it is not. The minimization can be formalized in the following manner:

$$\begin{aligned} \min \quad & cx \\ \text{subject to} \quad & \sum_{i=1}^{k} f_i(a_i x) \geq K \\ & x \in \{0, 1\}^n \end{aligned} \tag{1}$$

where $c = (c(T_1), \ldots, c(T_n))$ is the cost vector, K is the lower bound for the coverage, and a_i is the i^{th} row of the matrix A. Furthermore, let $v = (cov(P_1), \ldots, cov(P_k))$ be the coverage vector. Let us see how this formula looks like in the case of the three introduced models.

1. **Linear model**

$$\sum_{i=1}^{k} f_i(a_i x) = \sum_{i=1}^{k} \frac{a_i x}{b_i} cov(P_i) = \left(\sum_{i=1}^{k} \frac{a_i cov(P_i)}{b_i} \right) x$$

Thus (1) turns into a binary minimization with a single linear constraint:

$$\begin{array}{ll} \min & cx \\ \text{subject to} & \left(\sum_{i=1}^{k} \frac{a_i cov(P_i)}{b_i} \right) x \geq K \\ & x \in \{0,1\}^n \end{array} \tag{2}$$

2. **"All or nothing" model**
Let us introduce a new variable vector $z = (z_1, \ldots, z_k)$ defined in the following manner:

$$z_i = \begin{cases} 1 & \text{if } a_i x = b_i \\ 0 & \text{if } a_i x < b_i \end{cases}$$

In other words, $z = \max\{Ax - b + e, 0\}$, where $e = (1, 1, \ldots, 1)$ and the maximization is made componentwise. Using this vector problem (1) can be written in the following manner

$$\begin{array}{lrcl} \min & cx & & \\ \text{subject to} & vz & \geq & K \\ & z & = & \max\{Ax - b + e, 0\} \\ & x & \in & \{0,1\}^n \end{array}$$

It is easy to see that this is equivalent to

$$\begin{array}{lrcll} \min & cx & & & \\ \text{subject to} & z_i & \leq & \frac{a_i}{b_i} x & i = 1, \ldots, k \\ & z & \geq & Ax - b + e & \\ & vz & \geq & K & \\ & x & \in & \{0,1\}^n & \\ & z & \in & \{0,1\}^k & \end{array} \tag{3}$$

This is a binary minimization with linear constraints. The number of the variables is $n + k$, the number of the constraints is $2k + 1$.

3. **"One is enough" model**
This model can be handled similarly to the previous one. Let now z be the following vector:

$$z_i = \begin{cases} 0 & \text{ha } a_i x = 0 \\ 1 & \text{ha } a_i x \geq 1 \end{cases}$$

namely $z = \min\{Ax, e\}$. In this case (3) can be transformed into the following problem:

$$\begin{array}{rlll} \min & cx & & \\ \text{subject to} & z_i & \geq & \frac{a_i}{b_i}x \qquad i = 1, \ldots, k \\ & z & \leq & Ax \\ & vz & \geq & K \\ & x & \in & \{0,1\}^n \\ & z & \in & \{0,1\}^k \end{array} \tag{4}$$

Our second optimization problem is to find a **maximal coverage** test set supposing an upper bound for the cost (L). This optimization problem, as the previous one, can be formulated as a binary minimization problem with the functions f_i:

$$\begin{array}{rl} \max & \sum_{i=1}^{k} f_i(a_i x) \\ \text{subject to} & cx \leq L \\ & x \in \{0,1\}^n \end{array} \tag{5}$$

Without further details let us see the formulas for the three models.

1. **Linear model**

$$\begin{array}{rl} \max & \left(\sum_{i=1}^{k} \frac{a_i cov(P_i)}{b_i}\right) x \\ \text{subject to} & cx \leq L \\ & x \in \{0,1\}^n \end{array} \tag{6}$$

2. **"All or nothing" model**

$$\begin{array}{rlll} \max & vz & & \\ \text{subject to} & z_i & \leq & \frac{a_i}{b_i}x \qquad i = 1, \ldots, k \\ & z & \geq & Ax - b + e \\ & cx & \leq & L \\ & x & \in & \{0,1\}^n \\ & z & \in & \{0,1\}^k \end{array} \tag{7}$$

3. **"One is enough" model**

$$\begin{array}{rlll} \max & vz & & \\ \text{subject to} & z_i & \geq & \frac{a_i}{b_i}x \qquad i = 1, \ldots, k \\ & z & \leq & Ax \\ & cx & \leq & L \\ & x & \in & \{0,1\}^n \\ & z & \in & \{0,1\}^k \end{array} \tag{8}$$

4 The results of optimization

Having described the method, let us look at the experiments now. As it was mentioned in the introduction, we applied the method to the ISDN DSS1 Layer 2 protocol [2]. This TBR4 standard contains 27 test purposes (k=27) and 52 test cases (n=52) as well as the relation between the TCs and the TPs. The TCs that are necessary to check a TP are given for each purpose. Based on this standard the purpose-test incidence matrix can be easily constructed.

We fixed the coverage vector v in the value of $e = (1, \ldots, 1)$ because we did not want to distinguish the TCs with respect to the coverage. We defined three different cost vectors:

- In the first, $c_1(T_j) = 1$ for $j = (1, \ldots, n)$. This cost can be used if we are interested in only the number of the test cases; for example if their costs are all equal.
- The second cost vector (c_2) is based on the timers contained by the test cases. We estimated c_2 using the sum of the default times of the timers in the j^{th} test case. This time can be an upper bound for the execution time of T_j. The exact value of c_2 is as follows:

$$c_2 = (6,3,33,3,3,8,4,5,3,9,6,35,3,13,2,4,34,6,6,2,34,5,3,5,2,5,\\ 1,8,33,7,8,3,2,1,2,35,4,2,2,3,2,4,6,2,2,2,33,33,31,6,1,2)$$

- The definition of the third cost vector (c_3) is based on the assumption that the main time consuming steps of testing are the preparation and, in case of fault, the search for its cause. That is why we added a constant value (100) to every $c_2(T_j)$ referring to this time, so $c_3 = c_2 + 100$.

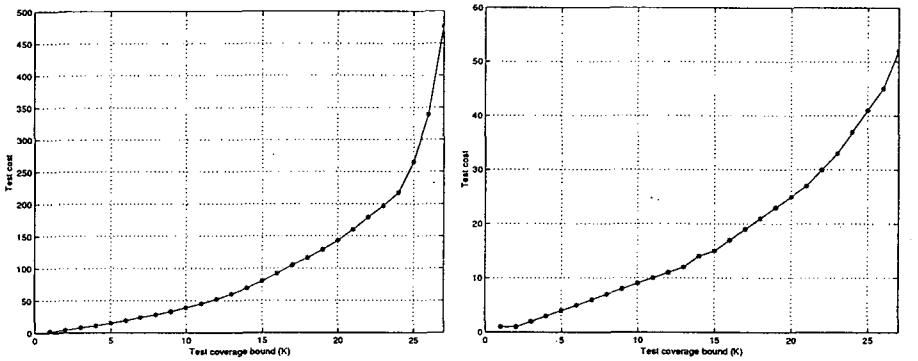
To solve the integer (binary) programming problems with the described parameters, we used the CPLEX program tool [5] .

4.1 Minimal costs

1. **Linear model**
   This model is less interesting than the others so we examined only the c_2 cost vector. The cost of the optimal test set for $K = 1, \ldots, 27$ is shown in Figure 2-a.

2. **"All or nothing" model**
   We examined all the three cost vectors. The results of the optimization problem (3) for $K = 1, \ldots, 27$ for the three cost functions are shown in Figure 2-b, Figure 3-a, Figure 3-b.

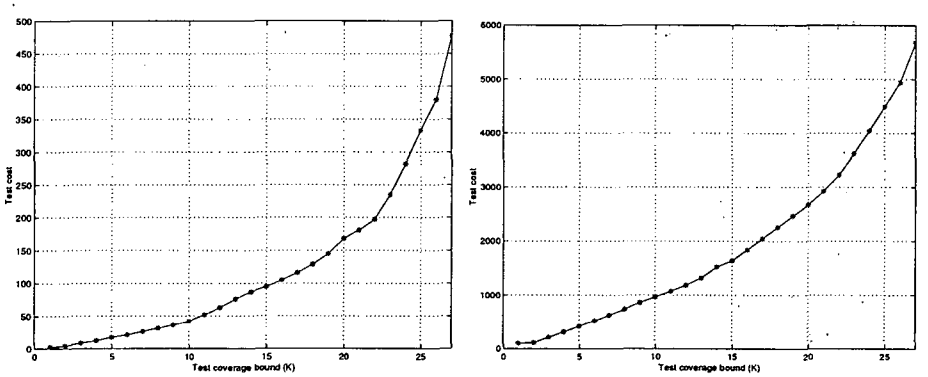
   We can see in all cases that increasing the coverage bound, the cost of the optimal test set does not increase linearly. This means that using our method we can obtain better (shorter in time) test sets to execute, than we would



(a) Linear model, c_2 cost vector

(b) "All or nothing" model, c_1 cost vector

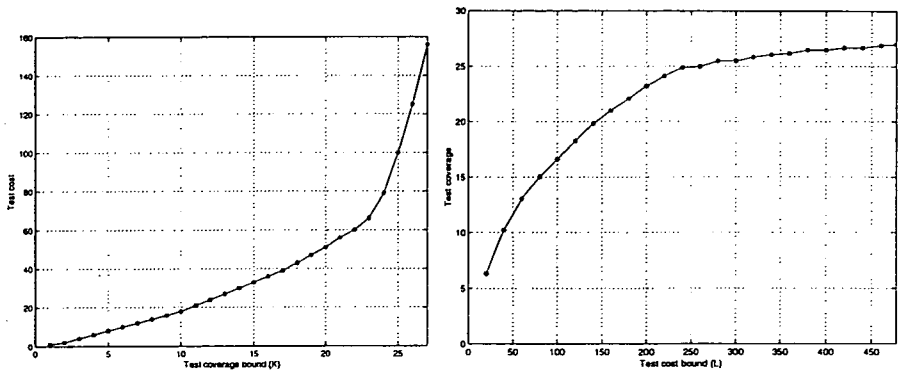
Figure 2: Minimal costs in Linear model and "All or nothing" model



(a) "All or nothing" model, c_2 cost vector

(b) "All or nothing" model, c_3 cost vector

Figure 3: Minimal costs in "All or nothing" model



(a) Minimal cost in "One is enough" model, c_2 cost vector

(b) Maximal coverage in Linear model, c_2 cost vector

Figure 4: Minimal cost in "One is enough" model and maximal coverage in Linear model

get if we chose at random. In fact, our method gives us the best possible test set within the constraints.

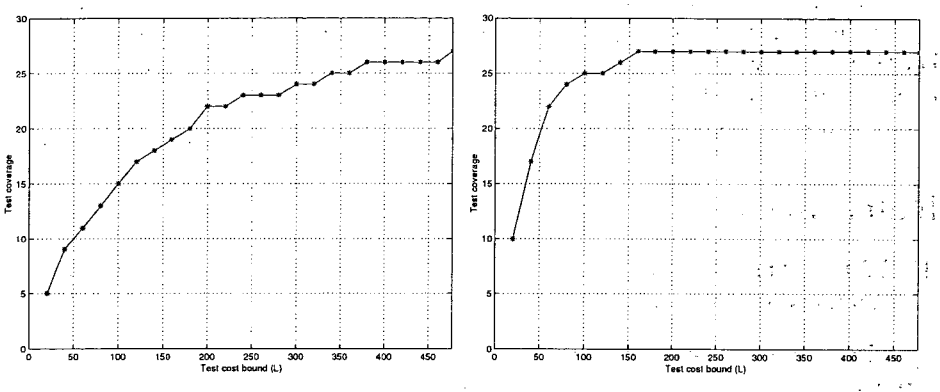
The figures also show that the biggest increase in the rate of the cost in coverage is in the case when the cost vector c_2 is used (Figure 3-a). This is because the variation of the cost values is the biggest in this case. That means we can reduce cost with only a small loss of test coverage. The cost jumps when, in order to reach the required test coverage, it is necessary to execute those test cases which have bigger cost values.

Where the variation of the cost vector is less, as in case c_3 and especially in c_1, the graph is smoother. This is quite logical as the execution of a given test case does not increase the total cost significantly regardless of which test case we select.

3. **"One is enough" model**
   Figure 4-a shows the optimal cost in the "One is enough" model using cost vector c_2

4.2 Maximal coverages

1. **Linear model** When we are looking for a maximal coverage test set we have an upper bound for the cost (L). Different cost vectors have different maximal upper bounds (L_{max} =52, 477 or 5677). We present only one graph for this model, Figure 4-b, which shows the results using cost vector c_2 and



(a) "All or nothing" model, c_2 cost vector (b) "One is enough" model c_2 cost vector

Figure 5: Maximal coverage in "All or nothing" model and "One is enough" model

$L_{max} = 477$.

2. **"All or nothing" model**
   Figure 5-a shows the result of the maximal coverage problem for "All or nothing" model using the cost vector c_2.

3. **"One is enough" model**
   The results of the maximal coverage problem for the "One is enough" model using cost vector c_2 are shown in Figure 5-b. We can observe that the graph reaches the highest cost value at $L = 160$, so in this model only the third of the total cost is enough for the whole coverage.

4.3 Conclusions

The reduction of the time or effort put into conformance testing while keeping the test coverage under control is very important for those who perform conformance testing. If the time of executing conformance testing is limited, then we have to select the more efficient test cases from the whole test suite in order to make testing possible within a shorter period of time.

Our method is based on selecting test cases from the ATS of a protocol, when we can select what portion of the test coverage we are willing to devote to shorten the time of test execution. To achieve minimal testing time for a given lower bound of coverage the method determines which test cases have to be selected for execution. The method is protocol independent, so it can be used in the testing of any protocol. In the ATS of some protocols, however, the incidence matrix looks different. If there is a one-to-one relation between the test purposes and test cases, then the method cannot give us usable results.

Since the first step in the application of our method is the construction of the incidence matrix, it determines if the method is well applicable to a given protocol. For those protocols the incidence matrix is a diagonal one, we are working on other approaches to be able to give selection criteria for the test cases based on the coverage and the time constraints.

References

[1] X.290-X.296 OSI conformance testing methodology and framework for protocol recommendations for ITU-T applications

[2] ETSI Draft prTBR4 Integrated Services Digital Network (ISDN); Attachment requirements for terminal equipment to connect to an ISDN using ISDN primary rate access

[3] S. T. Vuong, Jinsong Zhu, Jadranka Alilovic-Curgus, "Sensitivity analysis of the metric based test selection", IFIP TC6 10th Int. Workshop on Testing of Communication Systems, 1997.

[4] K. Tarnay, "Protocol Specification and Testing", Akadémiai Kiadó, 1991.

[5] CPLEX, Version 3.0, 1989-1994, CPLEX Optimization, Inc.

[6] T. Csöndes, B. Kotnyek, "A Mathematical Programming Method in Test Selection", EUROMICRO'97 Short Contribution Session, 1997.

[7] T. Csöndes, S. Dibuz, B. Kotnyek, "Conformance Testing of Communication Protocols", IFIP TC6 Int. Symposium on Automation and Informatics'97, 1997.

Acta Cybernetica **14** (1999) 239-250.

A Family of Fast Constant-Space Substring Search Algorithms

Harri Hakonen* and Timo Raita†

Abstract

This paper describes a new strategy for searching a substring in a given text. The method is based on the well-known Boyer–Moore algorithm complementing it with a technique called q-slicing, a form of probabilistic q-gram matching. As a result, we get a family of highly parametric algorithms apt for adaptation to the special properties inherent to the source which generates the input strings. The search procedure is independent of the alphabet size and appropriate for efficient and practical on-line implementations. Simulation results show that they are comparable to the fastest currently known Boyer–Moore variants.

1 Introduction

In the *string searching (pattern matching) problem* our task is to determine all positions in a given text, $text[1..n]$, where the pattern, $pat[1..m]$, occurs. This problem has been studied extensively (see e.g. [1] for a good survey) and several efficient and elegant solutions have been devised. The most efficient implementations, from the practical point of view, are based on the seminal ideas of Boyer and Moore [6]. The original Boyer–Moore algorithm (BM for short) aligns the pattern with a text position j, compares the corresponding symbols of $pat[1..m]$ and $text[j-m+1..j]$ starting from the last symbol of the pattern and advancing to the left. If a mismatch (if any) is found, the pattern is shifted forward with respect to the text and the process is repeated:

*Turku Centre for Computer Science (TUCS), e-mail: hat@cs.utu.fi

†Department of Computer Science, University of Turku, Lemminkäisenkatu 14 A, SF-20520 Turku, Finland, e-mail: raita@cs.utu.fi. The author acknowledges support by the Academy of Finland under grant No. 865431

```
Boyer–Moore search(pat[1..m], text[1..n])
    Preprocess pattern.
    while all text is not scanned do
(i)     Perform skip loop.
        if witness for a match is found then
(ii)        Perform match loop.
            if pattern is found then
                Report match.
            endif
        endif
(iii)   Shift pattern.
    endwhile
```

The search procedure consists of three distinct phases: (i) fast skipping over non-matching text regions, (ii) match checking when some evidence of a pattern occurrence has been found and (iii) shift to the next position. All these steps have been subject of refinements [7, 8, 10, 13, 15, 16]. A detailed analysis of the various BM substep combinations can be found in [10].

The power of the BM algorithm is largely based on a sophisticated strategy to move the pattern forward relative to the text in steps (i) and (iii). This is accomplished by forming two tables in $O(m)$ time prior to the actual search. The *match heuristic* table determines how much the pattern must be moved in order to realign the matched region of the text, $text[j-k..j]$, with an identical pattern substring also in the new position. For this, we need to determine the rightmost substring $pat[p-k..p]$, $p < m$, which is identical to the matched suffix $pat[m-k..m]$ (not overlooking the special case $0 < p < k$). In fact, when we find that $pat[m-k-1] \neq text[j-k-1]$, we know that in order to succeed at the next probe position, the condition $pat[p-k-1] = text[j-k-1]$ must also hold. Because of the large amount of space and time overhead, however, the match heuristic is usually not made so fine-grained. As a reasonable and quick approximation, it is only required that $pat[p-k-1] \neq pat[m-k-1]$. The *occurrence heuristic* table expresses the position of the rightmost occurrence of each symbol of the input alphabet Σ in the pattern. Thus, the occurrence heuristic determines, how much we can shift the pattern in order to align the mismatched text symbol with an identical pattern symbol. The length of the shift in step (iii) is then given by the maximum of the match and the occurrence heuristic values.

Let us consider the role and importance of the two heuristics. The well-known and widely used BM variant devised by Horspool [8] (BMH) discards the match heuristic due to its small significance with non-periodic patterns. This results in $O(mn)$ worst case complexity. However, on the average, the complexity is only $O(n/m)$, the same as that for the original BM method, implying that the BMH method is very fast in practice. Evidently, BMH makes shorter shifts (on the average) than BM because it uses less information. This behaviour is emphasized when σ, the size of the input alphabet, is small. On the other hand, the role of the

match heuristic becomes insignificant when σ becomes large (this is studied in more detail in [14]). As suggested in [13], we should try to compensate the omission of the match heuristic in other ways during the search. One alternative is to extend the occurrence heuristic for bigrams, incorporating thus both heuristics (at least partly) into one. This idea was introduced in [16] and was shown to give improved running times, especially for small input alphabets. Moreover, if we follow the idea of BMH and choose always (independently of the position where the mismatch occurred during the right to left scan) the bigram composed of the text symbols aligning with $pat[m-1]$ and $pat[m]$, we obtain a very close approximation to the match heuristic. In the special case, where the mismatch occurs at $pat[m-2]$, they are used identically. Thus, if we can generalize the approach (as suggested in [2, 3, 12]) and use q-grams ($q > 2$) of arbitrary length, we obtain a heuristic which is a hybrid of the two original ones. However, the disadvantage of the approach is that both the preprocessing time and the space demand increase rapidly, being proportional to σ^q. In Section 2 we show how this can be avoided by retrieving only the most important information scattered around the current text position and storing it into a compact unit. After this, we give an intuitive analysis of the selection strategy which maximizes the length of the shift. Simulation results of the new search method are also given in Section 3. Concluding remarks are presented in Section 4.

2 The q-slicing method

During the search, the pattern $pat[1..m]$ is always aligned with a substring $text[j-m+1..j]$, where j is the *current text position* ($m \leq j \leq n$). Thus, when we say that the current position is increased, it means that the pattern, which is considered to be positioned above the text, is shifted forward relative to the text. The information, on the basis of which the shift is made, is typically gathered from the text region $text[j-m+2..j]$; the symbol $text[j-m+1]$ does not contribute any information, because the length of the shift is always at least one. To increase the average shift length, the symbol $text[j+1]$ can also be used [15]. In the sequel, the text symbols which are used as the basis for the shift are defined by a *template* $\tau = \langle t_1, \ldots, t_q \rangle$ containing a strictly increasing sequence of integers t_k. Each t_k is an offset from the current text position j. Thus, the symbol $text[j+t_k]$ aligns with the pattern symbol $pat[m+t_k]$, iff $-m+1 \leq t_k \leq 0$. Otherwise ($1 \leq t_k \leq n-j$), the text symbol does not reside 'under' the pattern. Clearly, the elements of the template can have a chance to push the pattern forward only, if the condition $t_{k+1} - t_k \leq m$ ($k = 1, \ldots, q-1$) holds. Figure 1(a) presents an example of a template giving four offsets.

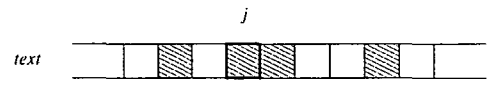
To give some insight into the efficiency of some commonly used templates, as well as some alternative choices, the following table summarizes the average shift lengths when a pattern of length 13 was searched for in an English book text (see

table on page 247 for more details of this input text).

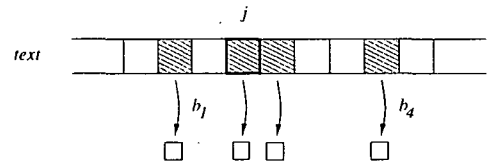
template τ	⟨0⟩	⟨-1,0⟩	⟨0,1⟩	⟨0,2⟩
average shift	9.65	12.53	13.44	13.72

template τ	⟨0,3⟩	⟨-2,-1,0⟩	⟨-1,0,1⟩	⟨0,1,2⟩
average shift	13.97	12.83	13.81	14.54

The text sampling processes defined by templates $\langle 0 \rangle$ and $\langle -1, 0 \rangle$ are used in the Horspool [8] and Zhu–Takaoka [16] algorithms, respectively. Also, $\langle 0, 1 \rangle$ can be seen as a generalization of Sunday's idea to exploit $text[j+1]$ in shifting [15]. The general tendency is clear: longer templates yield longer shifts. However, a carefully selected sampling strategy compensates short templates, as can be seen e.g. from the figures for $\langle 0, 3 \rangle$ and $\langle -2, -1, 0 \rangle$.



(a) template $\tau = \langle -2, 0, 1, 4 \rangle$



(b) selective mappings $\mu = \langle b_1(text[j-2]), b_2(text[j]), b_3(text[j+1]), b_4(text[j+4]) \rangle$

Figure 1: A structure of a 4-slice defined by $\tau : \mu = \langle -2 : b_1, 0 : b_2, 1 : b_3, 4 : b_4 \rangle$

In order to avoid the excessive time and space requirements during preprocessing and still achieve large shifts, we combine two ideas. First, the template is kept fixed during the search. The symbols indicated by the template are picked from the text at each probe position j. Second, we reduce the size of the alphabet at the cost of losing some accuracy in symbol comparison. The idea is to partition the symbols of the input alphabet Σ into equivalence classes and tag each class uniquely with a symbol of a reduced alphabet Σ' for which $\sigma' < \sigma$. Now, instead of comparing individual symbols, we compare tags of the classes. Thus, the approach is related to the 'shift-or' algorithm of Baeza-Yates and Gonnet [4], which can be used to search for substrings composed of metacharacters representing a set of symbols from the original alphabet. However, in the new scheme the tag is formed by mapping a symbol according to its position in the sampling template. In other words, the tag of $text[j + t_k]$ is the value of the corresponding function $b_k : \Sigma \mapsto \Sigma'$ $(1 \leq k \leq q)$.

The collection of these *selective mappings* is defined by the vector $\mu = \langle b_1, \ldots, b_q \rangle$. The combination of τ and μ is denoted by $\tau : \mu = \langle t_1 : b_1, \ldots, t_q : b_q \rangle$. Figure 1(b) shows an example of μ for the template $\tau = \langle -2, 0, 1, 4 \rangle$. Concatenation of the (reduced) symbols defined by the current text position j and the pair $\tau : \mu$ is called a *q-slice*:

$$q\text{-slice}_{\tau:\mu}(text, j) = \mathop{\mathtt{Concat}}_{i=1}^{q} \; b_i(text[j + t_i])$$

The q-slice scheme can be regarded as a hash function defined by τ giving the positions and μ representing the amount of information to be gathered. The equality of two slices is a necessary, but not a sufficient condition for the equality of the corresponding patterns [11]. The q-slice scheme introduces a general information sampling concept, and it has some interesting properties intrinsic to the string searching problem:

- The loss of information caused by the selective mappings is minimized, when the symbols of Σ' are uniformly distributed into the mapped sequences. Interestingly, by taking the three least significant bits from the (ASCII encoded) symbol representation has a nice property for natural languages: the most frequent symbols of the skew distribution fall into different equivalence classes. Because of this, and the fact that support for this type of operation can be found in most machine architectures, we restrict the form of the b_k functions $(k = 1, \ldots, q)$ in the sequel as follows:

  $$b_k(\alpha) = \alpha \;\; \mathtt{AND} \;\; 0 \ldots 011 \ldots 1, \quad \alpha \in \Sigma.$$

  The number of one-bits in the mask, lsb_k, determines how many least significant bits (LSBs) of the original symbol α we want to select. This specialized μ is called an *LSB-mask*. In what follows, each b_k function is denoted by the corresponding integer lsb_k. Naturally, other kinds of mappings are possible also, but they are not studied in this paper.

- Because the value of the hash function realized by the q-slice scheme is a concatenation of the mapped values, the function does not scramble the bits of the constituent symbols. Since important order information is preserved, it could be taken advantage of in the implementation of the search procedure: for each shift value calculated, we check whether the suffix of the previous hash value overlaps the prefix of the next one and in such a case, leads to a conflict. With a high probability this will happen, and we can increase the length of the shift. This improvement resembles the search strategy of Galil [7] and also the principles used in the construction of the BM match heuristic (cf. the description given in the Introduction). Although the proposed hash function is very simple, false matches occur rarely.

Collision probability of q-slice. Let $B = \log_2 \sigma$ and fix $B' = lsb_k$ for all $k = 1, \ldots, q$. Assuming that the text and the pattern have been generated by a uniform symbol distribution, the probability of a q-slice hash address collision is

$$P(q\text{-slice matches} \mid \text{pattern mismatches}) = \frac{2^{q(B-B')} - 1}{2^{qB} - 1}.$$

Proof. This follows from the probabilities $P(q\text{-slice matches}) = 2^{-qB'}$ and $P(\text{pattern mismatches} \mid q\text{-slice matches}) = 1 - 2^{-q(B-B')}$. ■

A simple but good approximation for this formula is the q-slice matching probability $2^{-qB'}$. For example, if $q = 2$, $B = 8$ and $B' = 3$, then $P(\text{collision}) \approx 0.0156$.

- The q-slice can be used efficiently for searching when its length, $lsb_1 + \ldots + lsb_q$, is conveniently chosen to fill a machine-dependent unit, e.g. a byte or a word, and the sampling strategy is supported by the architecture. Unfortunately, the latter is often true only for the trivial case $q = 1$. Therefore, this parameter is usually chosen to be small and a balanced intertwining of τ and μ becomes crucial. This is discussed in more detail in section 3.

The actual search procedure starts by preprocessing the pattern as follows. In the description below, we use the notation $\bullet_r | \bullet_{r+1} | \ldots | \bullet_s$ $(1 \leq r \leq s \leq q)$, when we want to refer to a part of a q-slice. Also, the set of all possible bit combinations of length lsb_k for the k'th component is denoted by $*_k$. For simplicity, we shall assume that there exists an index i for which $-m + 2 \leq t_i \leq 1$. Without this restriction, the algorithm would contain unnecessary details obscuring the basic idea; these special cases can be easily incorporated into the scheme by analysing them carefully (left as an easy exercise to the reader).

Define a proper template τ (of length q) and LSB-mask μ.
shiftLength$[*_1 | *_2 | \ldots | *_q] := m + t_q$
    // Each of the $2^{\sum_{i=1}^{q} lsb_i}$ table values is initialized to the maximal shift.
for $c := m + t_q - 1$ **downto** 1 **do**
    Find all t_k values in range $-m + 1 + c..m + c$.
        // These are characterized by indices r and s for which $t_r \leq t_k \leq t_s$.
        // Each t_k aligns with $pat[t_k - (m + t_q - c)]$ after a shift of
        // length c. Thus, we call these offsets *bound (template)*
        // *positions*. The others (i.e. $t_1, \ldots, t_{r-1}$ and $t_{s+1}, \ldots, t_q$)
        // are *free positions*.
    Determine the unique part of the q-slice $\bullet_r | \bullet_{r+1} | \ldots | \bullet_s$ corresponding to the bound positions.
    shiftLength$[*_1 | \ldots | *_{r-1} | \bullet_r | \ldots | \bullet_s | *_{s+1} | \ldots | *_q] := c$

// Make $2^{\sum_{i=1}^{r-1} lsb_i + \sum_{i=s+1}^{q} lsb_i}$ updates.

endfor

After preprocessing, the search is performed by mapping the text symbols to the reduced alphabet on-the-fly using the q-slice$_{r:\mu}(text, j)$. The slice contains information which is scattered into a large region near the current context. This gives a basis to increase the average length of a shift using only a small amount of comparisons. A minor drawback of the approach is, that when we encounter a q-slice match, we must confirm that an identical symbol pattern has been found.

Example. Let us assume that the symbols are ASCII encoded and that the pattern is $pat[1..14]$ = *abracadabracab*. The two least significant bits of the symbols '*a*', '*b*', '*c*', '*d*' and '*r*' are '01', '10', '11', '00' and '10', respectively. The template $\langle -1, 0, 1\rangle$ and the LSB-mask $\langle 2, 1, 1\rangle$ generate 16 different q-slices and their corrsponding shift lengths:

q-slice	00|0|0	00|0|1	00|1|0	00|1|1	01|0|0	01|0|1	01|1|0	01|1|1
shift	15	14	6	14	5	7	13	2

q-slice	10|0|0	10|0|1	10|1|0	10|1|1	11|0|0	11|0|1	11|1|0	11|1|1
shift	15	4	13	3	15	14	1	14

For example, the shift value for the q-slice 10|0|1 is 4 because $pat[10..12] = rac$ is the rightmost pattern region that matches it. Obviously, the shift values are always in the range $1..m + t_q$. Assuming that the pattern and text are aligned as in (a):

pat ... *a c a b*
text ... *a c a d b a r b a c* ...

(a)

pat ... *a d a b r a c a b*
text ... *a c a d b a r b a c* ...

(b)

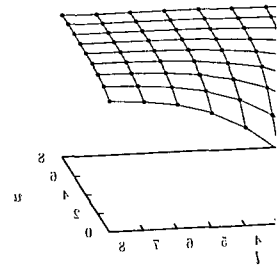
we find that q-slice 01|0|0 obtained from the text symbols a, d, b tells us to shift the pattern 5 positions forward (as shown in (b)) in order to align *abr* with *adb*. No pattern occurrence is found here either, and the template symbols b, a, c generate the q-slice 10|1|1 yielding a new shift of length 3.

3 Experiments

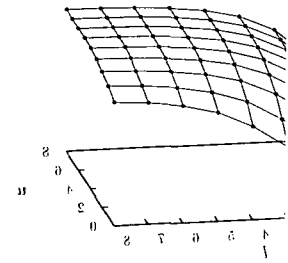
Expectation of the shift length. The basic structure of the BM algorithm, given in the introduction, shows that before the pattern shift is made in step (iii), we have gathered a lot of information about the symbols near the current text position.

In this experimental analysis, however, we assume that no such information is available. To obtain a realiable comparative analysis and at the same time keep the various parameter combinations practically feasible, we restrict ourselves to a 2-slice of type $\tau : \mu = \langle 0 : \ell, (D+1) : u \rangle$. In other words, the pattern is shifted according to the ℓ-bit tag of $text[j]$ and u-bit tag of $text[j + D + 1]$. The ℓ bits are extracted from the text symbol which resides under the ℓast pattern symbol. This information produces an 'initial shift' for the pattern. The $(D+1) : u$ component gives an 'additional push', since it probes further information from an $u$pcoming text symbol at the distance D from the current text position. This special algorithm family is denoted by $\ell \lhd D \rhd u$, where $\lhd \cdots \rhd$ symbolizes the distance between the two units from which the bits are extracted.

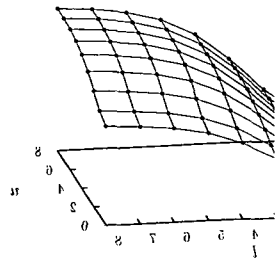
Let us study the expectation of the shift lengths for the $\ell \lhd D \rhd u$ algorithm when $m = 13$; $\ell, u = 0, \ldots, 8$ and $D = 0, 1, m/2$. To analyze the behaviour of the expectation for natural languages, a simulation over an English book text (see table on page 247) was accomplished. The search was performed for 30 randomly selected patterns from the text. Figure 2 shows the expectation of the shift length based on this test arrangement.



(a) $D = 0$



(b) $D = 1$



(c) $D = 6$

Figure 2: The expectation of the shift length for $\ell \lhd D \rhd u$ algorithms

The expectation behaves differently for the following two cases.

Case $D = 0$. The average shift values are almost symmetrical wrt the diagonal $\ell = u$ (Fig 2(a)). For the region $\ell + u \geq 6$, the expected shift length is always ≥ 12. This suggests that even a shift table of size 64 gives a good performance for English text.

Case $D \geq 1$. The shape of the expectation function differs from the previous case: whenever $\ell = 0$, we have now no information to use any other shift length except 1. The influence of the parameter ℓ is more significant than that of u and only with parameter values $\ell + u \geq 7$ constrained by $\ell \geq 4$, we reach the average shift of at least 12 positions (Fig 2(b,c)). Referring back to figure 2(a), we can observe that if $u = 0$ and ℓ approaches the length of the original encoding of $text[j]$, we quite quickly reach the situation where the shifts are larger than $m/2$. Comparing this with the results of Fig2(c), we can see how much more the upcoming symbol is able push the pattern forward once the initial push has been given. This also explains why we can have an average shift length which is significantly larger than m, the length of the pattern.

Running times. After extensive test runs, we suggest the schemes $4 \lhd 0 \rhd 2$ and $3 \lhd 0 \rhd 3$ as general purpose substring searching algorithms. Evidently, if the properties of the input strings differ significantly from those of English, some other parameter values may result in better performance. Furthermore, the implementations of the fastest currently known search algorithms are extremely carefully designed and the hardware architecture may have a large effect on their speed.

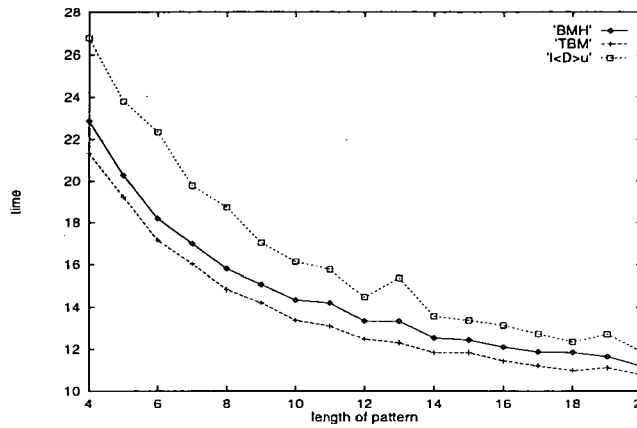
The $4 \lhd 0 \rhd 2$ algorithm was tested and compared to the basic Horspool variant BMH [8] and to the Hume–Sunday variant TBM [10]. To our knowledge, TBM is one of the fastest, widely known algorithms for natural language text search. Tests were run on a Sparc machine (architecture `sun4m`, kernel `SunOS 5.6`) and the C programs were compiled with `gcc` (version 2.7.2.3) using the optimization switch `-O3`.

The input data consisted of the English book text and a dna text, having the following properties.

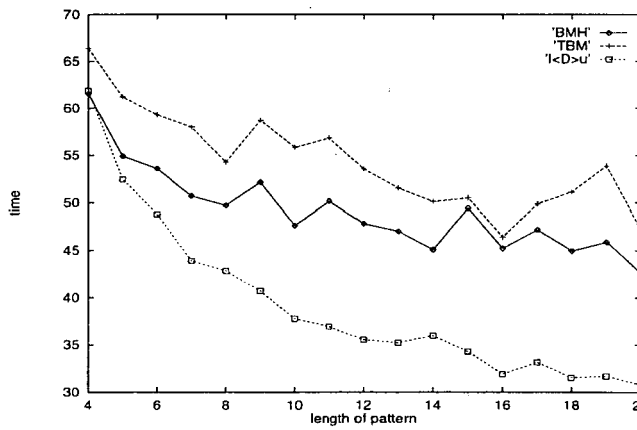
text type	source	used file	σ	length
English book	Calgary Corpus [5]	`book2`	96	611 kB
Dna sequence	[10]	`dna.test`	4	988 kB

The simulation was accomplished by selecting 30 patterns of length m randomly in the text and then searching for all of their occurrences in the text. This was repeated for $m = 4, \ldots, 20$. To make the comparison fair, the running times include

both searching and preprocessing phases. Figure 3 shows the results of this test set.



(a) English text



(b) Dna sequence

Figure 3: The running times of BM, BMH, $4 \lhd 0 \rhd 2$ and $2 \lhd 0 \rhd 2$ algorithms

The running times of the $4 \lhd 0 \rhd 2$ algorithm are quite modest for natural languages (see Fig 3(a)). This is due to the hardware architecture, which does not support multicharacter sampling. However, the shape of the curve of the new scheme shows that the information obtained from a q-slice is at least as good as if we used more local, but exact information. When the size of the input alphabet is decreased, the 'traditional' methods begin to lose their power because the machine-level size of a symbol unit is typically kept fixed although the information content of a unit is

smaller. This deficiency is handled in the new method, as it gathers and utilizes data of size q extracted from the neighbourhood of the current position. The effect can be seen strikingly in Fig. 3(b): the form of the curve for the q-slice method remains identical to that for large alphabets whereas the smoothness of the performance for the two other methods disappears. Moreover, it is not only the $O(n/m)$ behaviour which is lost but BMH and TBM are also clearly much slower than the new method. Since $\sigma = 4$ for dna sequences, $2 \lhd 0 \rhd 2$ was chosen as the representative of the new approach (Fig 3(b)). As a final remark, recall that the running times of the $\ell \lhd D \rhd u$ algorithms are independent of Σ, unlike the BMH and TBM methods.

4 Summary

A new family of fast substring searching algorithms using q-slices is devised. The concept of a q-slice combines the idea of using q-grams together with the mapping of symbols to a reduced alphabet. This new strategy makes on-line text sampling to skip fast over regions where the pattern cannot occur. In spite of the fact that most machine architectures do not support the core operation of q-slicing on the hardware level, the efficiency of the new method is comparable to the fastest known substring search algorithms. Tests have shown that this approach typically results in average shift lengths which are even larger than the size of the pattern. This algorithm family is highly parametric and can thus easily be adapted to specific application environments when necessary.

References

[1] Baeza-Yates, R.A.: Algorithms for String Searching: A Survey, *SIGIR Forum, Spring/Summer 1989, Vol. 23, No. 3,4, pp. 34-58*

[2] Baeza-Yates, R.A.: Improved String Searching, *Softw. Pract. Exp., Vol. 19, No. 3, March 1989, pp. 257-271*

[3] Baeza-Yates, R., Krogh, F.T., Ziegler, B., Sibbald, P.R. & Sunday, D.M.: Notes on a Very Fast Substring Search Algorithm, *Comm. ACM, Vol. 35, No. 4, April 1992, pp. 132-137*

[4] Baeza-Yates, R.A. & Gonnet, G.H.: A New Approach to Text Searching, *Proc. of the SIGIR Conference 1989, pp. 168-175*

[5] Bell, T.C., Cleary, J.G. & Witten, I.H.: Text Compression, *Prentice-Hall, 1990*

[6] Boyer, R.S. & Moore, J.S.: A Fast String Searching Algorithm, *Comm. ACM, Vol. 20, No. 10, October 1977, pp. 762-772*

[7] Galil, Z.: On Improving the Worst Case Running Time of the Boyer–Moore String Matching Algorithm, *Comm. of the ACM, Vol. 22, No. 9, September 1979, pp. 505-508*

[8] Horspool, R.N.: Practical Fast Searching in Strings, *Softw. Pract. Exp., Vol. 10, 1980, pp. 501-506*

[9] Galil, Z. & Seiferas, J.: Time-Space-Optimal String Matching, *J. Comput. System Sci., Vol. 26, 1983, pp. 280-294*

[10] Hume, A. & Sunday, D.M.: Fast String Searching, *Softw. Pract. Exp., Vol. 21, No. 11, November 1991, pp. 1221-1248*

[11] Karp, R.M. & Rabin, M.O.: Efficient Randomized Pattern-matching Algorithms, *IBM J. Res. Develop., Vol. 31, No. 2, March 1987, pp. 249-260*

[12] Knuth, D.E., Morris, J.H. & Pratt, V.R.: Fast Pattern Matching in Strings, *Siam J. Comput., Vol. 6, No. 2, June 1977, pp. 323-350*

[13] Raita, T.: Tuning the Boyer–Moore–Horspool String Searching Algorithm, *Softw. Pract. Exp., Vol. 22, No. 10, October, 1992, pp. 879-884*

[14] Tarvainen, H.: A Theoretical Framework for the Substring Searching Algorithms, *M.Sc. Thesis (in Finnish), University of Turku, Finland, May 1995*

[15] Sunday, D.M.: A Very Fast Substring Search Algorithm, *Comm. of the ACM, Vol. 33, No. 8, August 1990, pp. 132-142*

[16] Zhu, R.F. & Takaoka, T.: On Improving the Average Case of the Boyer–Moore String Matching Algorithm, *J. of Inf. Proc., Vol. 10, No. 3, 1987, pp. 173-177*

Acta Cybernetica 14 (1999) 251-261.

On a Merging Reduction of the Process Network Synthesis Problem*

Cs. Holló †, Z. Blázsik ‡ Cs. Imreh †, Z. Kovács †

Abstract

Since the combinatorial version of the process network synthesis (PNS) problem is NP-complete, it is important to establish such methods which render possible the reduction of the size of model. In this work, a new method called merging reduction is introduced which is based on the merging of operating units. The mergeable operating units are determined by an equivalence relation on the set of the operating units, and all of the operating units included in an equivalence class are merged into one new operating unit. This reduction has the following property: an optimal solution of the original problem can be derived from an optimal solution of the reduced problem and conversely. Presentation of this reduction technique is equipped with an empirical analysis on randomly generated problems which shows the measure of the size decrease.

1 Preliminaries

The foundations of PNS and the background of the combinatorial model studied here can be found in [3], [4], [5], and [9]. Therefore, we shall confine ourselves only to the recall of the definitions here. The merging reduction is presented in Section 2, while Section 3 contains the results of our empirical analysis.

In the combinatorial approach, the structure of a process can be described by the process graph (see [4]) defined as follows.

Let M be a finite nonempty set, the set of the materials. Furthermore, let $\emptyset \neq O \subseteq \wp'(M) \times \wp'(M)$ with $M \cap O = \emptyset$ where $\wp'(M)$ denotes the set of all nonempty subsets of M. The elements of O are called *operating units* and for an operating unit $(\alpha, \beta) \in O$, α and β are called the *input-set* and *output-set* of the operating unit, respectively. Pair (M, O) is defined to be a *process graph* or P-*graph* in short. The set of vertices of this directed graph is $M \cup O$, and the set

*This work has been supported by the Ministry of Culture and Education of Hungary, Grant FKFP 0008/1999 and by the Hungarian National Foundation for Scientific Research, Grant TO 30074.

†Department of Informatics, József Attila University, Árpád tér 2, H-6720 Szeged, Hungary

‡Research Group on Artificial Intelligence, Hungarian Academy of Sciences, Aradi vértanúk tere 1, H-6720 Szeged, Hungary

of arcs is $A = A_1 \cup A_2$ where $A_1 = \{(X,Y) : Y = (\alpha,\beta) \in O \text{ and } X \in \alpha\}$ and $A_2 = \{(Y,X) : Y = (\alpha,\beta) \in O \text{ and } X \in \beta\}$. If there exist vertices $X_1, X_2, \ldots, X_n$, such that $(X_1,X_2),(X_2,X_3),\ldots,(X_{n-1},X_n)$ are arcs of process graph (M,O), then the path determined by these arcs is denoted by $[X_1,X_n]$.

Now, let $o \subseteq O$ be arbitrary. Let us define the following functions on set o

$$mat^{in}(o) = \bigcup_{(\alpha,\beta)\in o} \alpha, \qquad mat^{out}(o) = \bigcup_{(\alpha,\beta)\in o} \beta,$$

and

$$mat(o) = mat^{in}(o) \bigcup mat^{out}(o).$$

Let process graphs (m,o) and (M,O) be given. (m,o) is defined to be a *subgraph* of (M,O), if $m \subseteq M$ and $o \subseteq O$.

Now, we can define the structural model of PNS for studying the problem from structural point of view. For this reason, let M^* be an arbitrarily fixed possibly infinite set, the set of the available materials. By *structural model* of PNS, we mean a triplet (P,R,O) where P, R, O are finite sets, $\emptyset \neq P \subseteq M^*$ is the set of the *desired products*, $R \subseteq M^*$ is the set of the *raw materials*, and $O \subseteq \wp'(M^*) \times \wp'(M^*)$ is the set of the available operating units. It is assumed that $P \cap R = \emptyset$ and $M^* \cap O = \emptyset$, furthermore, α and β are finite sets for every $(\alpha,\beta) = u \in O$.

Then, process graph (M,O), where $M = \cup\{\alpha \cup \beta : (\alpha,\beta) \in O\}$, presents the interconnections among the operating units of O. Furthermore, every feasible process network, producing the given set P of products from the given set R of raw materials using operating units from O, corresponds to a subgraph of (M,O). Examining the corresponding subgraphs of (M,O), therefore, we can determine the feasible process networks. If we do not consider further constraints such as material balance, then the subgraphs of (M,O) which can be assigned to the feasible process networks have common combinatorial properties. They are studied in [4] and their description is given by the following definition.

Subgraph (m,o) of (M,O) is called a *solution-structure* of (P,R,O) if the following conditions are satisfied:

(A1) $P \subseteq m$,

(A2) $\forall X \in m$, $X \in R \Leftrightarrow$ no (Y,X) arc in the process graph (m,o),

(A3) $\forall Y_0 \in o$, $\exists$ path $[Y_0,Y_n]$ with $Y_n \in P$,

(A4) $\forall X \in m$, $\exists(\alpha,\beta) \in o$ such that $X \in \alpha \cup \beta$.

The set of solution-structures of $\mathbf{M} = (P,R,O)$ will be denoted by $S(P,R,O)$ or $S(\mathbf{M})$.

Let us consider PNS problems in which each operating unit has a weight. We are to find a feasible process network with the minimal weight where by weight of a process network we mean the sum of the weights of the operating units belonging to the process network under consideration. Each feasible process network in such

a class of PNS problems is determined uniquely from the corresponding solution-structure and vice versa. Thus, the problem can be formalized as follows:

PNS problem with weights

Let a structural model of PNS problem $\mathbf{M} = (P, R, O)$ be given. Moreover, let w be a positive real-valued function defined on O, the weight function. The basic model is then

$$\min\{\sum_{u \in o} w(u) : (m, o) \in S(P, R, O)\}. \tag{1}$$

It is known (see [1],[2], and [10]) that this problem is NP-complete. In what follows, for the sake of simplicity, we call the elements of $S(\mathbf{M})$ *feasible solutions* and by PNS problem we mean a PNS problem with weights.

It is a basic observation that if (m, o) and (m', o') are solution-structures of $\mathbf{M}$, then $(m, o) \cup (m', o')$ is also a solution-structure of $\mathbf{M}$. This yields that $S(\mathbf{M})$ has a greatest element called *maximal structure* provided that $S(\mathbf{M}) \neq \emptyset$. Indeed, the maximal structure is the union of all the solution-structures of $\mathbf{M}$. Obviously, the P-graph of an arbitrary PNS problem can contain unnecessary operating units and materials. On the basis of the maximal structure, we can disregard from these unnecessary operating units and materials as follows. Let $(\bar{M}, \bar{O})$ denote the P-graph of the maximal structure. Then, the P-graph of structural model $\bar{\mathbf{M}} = (P, R \cap \bar{M}, \bar{O})$ is $(\bar{O}, \bar{M})$, and since each solution-structure of $\mathbf{M}$ is a subgraph of $(\bar{M}, \bar{O})$, it is a solution-structure of $\bar{\mathbf{M}}$, and conversely. Consequently, $S(\mathbf{M}) = S(\bar{\mathbf{M}})$. On the other hand, $\bar{\mathbf{M}}$ does not contain any unnecessary operating unit and material. Structural model $\bar{\mathbf{M}}$ is called *reduced structural model* of PNS.

To determine the reduced structural model for a PNS problem, an effective procedure is presented in [6], [7]; it can decide if $S(\mathbf{M})$ is empty; if $S(\mathbf{M})$ is not empty, the algorithm provides the corresponding maximal structure. Regarding the significance of this reduction, an empirical analysis is presented in [11], where the reduction procedure is executed on randomly generated PNS problems. It turned out that the decrease of size is about 47%.

Now, we recall this algorithm. This procedure consists of two major parts. The first part is intended to reduce the set of available operating units by eliminating some or all inappropriate operating units. Even if one desired product cannot be generated by any of the remaining operating units, no solution-structure exists for the structural model of PNS under consideration; consequently, there is no maximal structure. If it is still possible to have the maximal structure, then the second part of the algorithm constructs a P-graph from a subset of the operating units left after the first part, which is exactly the maximal structure. To elucidate this procedure, let a structural model of PNS be given by $\mathbf{M} = (P, R, O)$.

Algorithm for Maximal Structure Generation

1. Reduction

Initialization

- Let $O_0 = O \setminus \{(\alpha,\beta) : (\alpha,\beta) \in O \ \& \ \beta \cap R \neq \emptyset\}$ and $M_0 = mat(O_0)$. If $P \not\subseteq M_0$, then terminate since there is no maximal structure for **M**. If not, then let $T_0 = \{X : X \in M_0 \setminus R \ \& \ ((\alpha,\beta) \in O_0 \longrightarrow X \notin \beta)\}$. Finally, set $r := 0$.

Iteration

- *Step 1.1.* If $T_r = \emptyset$, then proceed to the initialization for building. If not, then choose a material X from T_r and set $O_X = \{(\alpha,\beta) : (\alpha,\beta) \in O_r \ \& \ X \in \alpha\}$. Let $O_{r+1} = O_r \setminus O_X$; moreover, $M_{r+1} = mat(O_{r+1})$. If $P \not\subseteq M_{r+1}$, then terminate since there is no maximal structure for **M**. If not, then construct set T'_r by $T'_r = \{Y : Y \in mat^{out}(O_X) \ \& \ Y \notin mat^{out}(O_{r+1}) \ \& \ Y \in mat^{in}(O_{r+1})\}$. Let $T_{r+1} = (T_r \cap M_{r+1}) \cup T'_r$. Set $r := r + 1$ and proceed to the following iteration for reduction.

2. Building

Initialization

- Let $W_0 = P$, $m_0 = \emptyset$ and $o_0 = \emptyset$; moreover, set $s := 0$.

Iteration

- *Step 2.1.* If $W_s = \emptyset$, then terminate. There exists at least one solution-structure for **M**. In particular, $(\bar{m}, o_s)$ is the maximal structure of **M** where $\bar{m} = mat(o_s)$. If $W_s \neq \emptyset$, then proceed to Step 2.2.

- *Step 2.2.* Choose one material from W_s; denote this material by X, and let $m_{s+1} = m_s \cup \{X\}$. Then, form set $O^*_X = \{(\alpha,\beta) : (\alpha,\beta) \in O_r \ \& \ X \in \beta\}$. Also, let $o_{s+1} = o_s \cup O^*_X$ and $W_{s+1} = (W_s \cup mat^{in}(O^*_X)) \setminus (R \cup m_{s+1})$. Then, set $s := s + 1$, and proceed to the succeeding iteration for building.

2 Merging reduction

While the general reduction presented above renders possible to exclude the unnecessary operating units and materials from the investigation, the merging reduction compresses the P-graph by merging some of its operating units. If $u_1 = (\alpha_1, \beta_1)$ and $u_2 = (\alpha_2, \beta_2)$, then one can merge these two operating units into a new operating unit defined by $u = (\alpha_1 \cup \alpha_2, \beta_1 \cup \beta_2)$. It is worth noting that after the merging of two or more operating units, we obtain a new structural model of PNS.

If we want to use this new structural model for solving the original problem, then a strong relationship must be established between the feasible solutions of the two problems. To establish this relationship, it is a basic question that which operating units are mergeable.

For this purpose, let $\mathbf{M} = (P, R, O)$ be a reduced structural model of PNS. Then, operating units $u_1, u_2 \in O$ are called *mergeable* if for any feasible solution, either both of them are contained in it or both of them are excluded from it. Formally stated, u_1 and u_2 are mergeable if $u_1 \in o$ implies $u_2 \in o$, and conversely, for every feasible solution $(m, o) \in S(\mathbf{M})$.

It can be readily seen that this relation is reflexive, symmetric, and transitive, and thus, it is an equivalence relation on set O which is denoted by $\equiv$. Let us define structural model $\mathbf{M}/\equiv = (P, R, O^*)$ by

$$O^* = \{(\cup\{\alpha_t : u_t = (\alpha_t, \beta_t) \in C(u)\}, \cup\{\beta_t : u_t = (\alpha_t, \beta_t) \in C(u)\}) : u \in O\}$$

where $C(u)$ denotes the equivalence class containing u. The visual meaning of $\mathbf{M}/\equiv$ can be given as follows. For each equivalence class, we merge all of the operating units belonging to this class into a new operating unit. This new operating unit will substitute the original ones in $\mathbf{M}/\equiv$. Obviously, $\mathbf{M}/\equiv$ is a structural model of PNS and its maximal structure is (M, O^*). Now, we define a mapping φ of $M \cup O$ onto $M \cup O^*$. For every $X \in M$, let $\varphi(X) = X$, furthermore, for every $u_s \in C(u)$, let $\varphi(u_s) = (\cup\{\alpha_t : u_t \in C(u)\}, \cup\{\beta_t : u_t \in C(u)\})$. As it is usual, we shall use the notation $\varphi(o) = \{\varphi(u) : u \in o\}$ and $\varphi(m) = \{\varphi(X) : X \in m\}$ for a subset o of O and for a subset m of M, respectively. Using this extension, we can take the image of an arbitrary P-graph (m, o) of (M, O) under φ as $(\varphi(m), \varphi(o))$. This mapping is denoted also by φ.

The following statement establishes a strong relationship between the two sets $S(\mathbf{M})$ and $S(\mathbf{M}/\equiv)$ of feasible solutions.

Theorem 1. *Mapping φ is a bijective mapping of $S(\mathbf{M})$ onto $S(\mathbf{M}/\equiv)$.*

Proof. Let $(m, o) \in S(\mathbf{M})$ be an arbitrary feasible solution. First, it is shown that $(\varphi(m), \varphi(o))$ is a feasible solution of $\mathbf{M}/\equiv$. Obviously, $(\varphi(m), \varphi(o))$ is such a P-graph which is a subgraph of (M, O^*). Consequently, it is enough to prove that $(\varphi(m), \varphi(o))$ satisfies conditions $(A1)$ through $(A4)$. Condition $(A1)$ is clearly valid, since $P \subseteq m = \varphi(m)$. To prove condition $(A2)$, let us observe that mapping φ preserves the sources. Regarding condition $(A3)$, let $u \in \varphi(o)$ be an arbitrary operating unit. Then, there is at least one $u_j \in o$ such that $\varphi(u_j) = u$. On the other hand, $(m, o) \in S(\mathbf{M})$, and thus, on the base of $(A3)$, there is a $[u_j, Y_n]$ path in (m, o) with $Y_n \in P$. Now taking the images under φ of the vertices of this path, we obtain a path $[u, Y_n']$ in $(\varphi(m), \varphi(o))$ where $Y_n' \in P$ which implies the validity of $(A3)$. Finally, to prove $(A4)$, let $X \in \varphi(m)$ be arbitrary. Then, $X \in m$, and by property $(A4)$, there exists an operating unit $u_j = (\alpha_j, \beta_j)$ such that $X \in \alpha_j \cup \beta_j$. Let $\varphi(u_j) = (\alpha, \beta)$. Then, by the definition of $\varphi(u_j)$, $X \in \alpha \cup \beta$, thereby validating $(A4)$.

Now, it is proven that φ is an injective mapping. For this purpose, let $(m,o) \neq (m',o') \in S(\mathbf{M})$. If $m \neq m'$, then $\varphi(m) \neq \varphi(m')$, and thus, the images are different. Otherwise, $o \neq o'$. To prove this case by contradiction, let us suppose that $(\varphi(m),\varphi(o)) = (\varphi(m'),\varphi(o'))$. Since $o \neq o'$, without loss of generality, we may assume that there exists a $u' \in o'$ with $u' \notin o$. Let $\varphi(u') = u$. Since $(\varphi(m),\varphi(o)) = (\varphi(m'),\varphi(o'))$, there exists a $\bar{u} \in o$ with $\varphi(\bar{u}) = u$. Then, by the definition of φ, $\bar{u} \equiv u'$, and thus, by the definition of the equivalence relation, $u' \in o$ which is a contradiction. Consequently, φ is a one-to-one mapping.

Finally, we show that φ is a mapping of $S(\mathbf{M})$ onto $S(\mathbf{M}/\equiv)$. For this purpose, let us consider an arbitrary feasible solution denoted by (m^*,o^*) of $S(\mathbf{M}/\equiv)$. Let $m = m^*$ and $o = \{u_j : u_j \in O \ \& \ \varphi(u_j) \in o^*\}$. Obviously, $\varphi(m,o) = (\varphi(m),\varphi(o)) = (m^*,o^*)$. Therefore, we have to prove that (m,o) is a feasible solution of $\mathbf{M}$. It can be easily seen that (m,o) is such a P-graph which is a subgraph of (M,O). Thus, we have to prove that (m,o) satisfies conditions $(A1)$ through $(A4)$.

Since $(m^*,o^*) \in S(\mathbf{M}/\equiv)$, condition $(A1)$ implies $P \subseteq m^*$. On the other hand, $m = m^*$, thereby indicating the validity of $(A1)$ for (m,o).

Since the ancestor of a source in (m^*,o^*) is a source in (m,o) under φ, and (m^*,o^*) satisfies $(A2)$, (m,o) satisfies condition $(A2)$ as well.

To prove $(A3)$ by contradiction, let us suppose that $(A3)$ is not valid for (m,o). Let us denote by o_1 the set of operating units in o from which there is no path in (m,o) into some required product, *i.e.*, let

$$o_1 = \{u_j : u_j \in o \ \& \ \text{no } [u_j,Y] \text{ path exists with } Y \in P \text{ in } (m,o)\}.$$

By our assumption, $o_1 \neq \emptyset$. Now, let us consider P-graph (m',o') where $o' = o \setminus o_1$ and $m' = mat(o')$. We shall prove that (m',o') is a feasible solution of $\mathbf{M}$.

Since $(m^*,o^*) \in S(\mathbf{M}/\equiv)$, $(A1)$ implies that for any $X \in P$, there exists an operating unit u producing X directly. Taking an ancestor of u, we obtain that there is an operating unit denoted by u' in o producing X directly, and thus, u' is not contained in o_1. Consequently, $u' \in o'$, thereby resulting in $P \subseteq m'$, *i.e.*, (m',o') satisfies condition $(A1)$.

To prove $(A2)$, let $X \in m'$ be arbitrary. If $X \in R$, then X is a source in (m^*,o^*), and since the ancestor of X is a source in (m,o) under φ, X is a source of (m,o). But $(m',o') \subseteq (m,o)$, and thus, X is a source in (m',o'). Conversely, let us suppose that X is a source in (m',o'). Then, X is a source in (m,o). Indeed, in the opposite case, X would be an output material of at least one operating unit from o_1. Let u_1 denote such an operating unit. Then, there is a $[u_1,Y]$ path in (m,o) since X is a source in (m',o'), and thus, X is an input material for some operating unit in o'. This fact contradicts the definition of o_1. Hence, X is a source in (m,o). In this case, X is a source in (m^*,o^*), and since $(A2)$ is valid for (m^*,o^*), $X \in R$. Consequently, (m',o') satisfies $(A2)$.

The validity of conditions $(A3)$ and $(A4)$ follows from the definitions of o_1 and (m',o'), and thus, we obtain that (m',o') is a feasible solution of $\mathbf{M}$.

Now, let us observe that $\varphi(m', o') = (m^*, o^*) = \varphi(m, o)$, which implies $(m', o') = (m, o)$ since φ is injective. Hence, $o_1 = \emptyset$ which is a contradiction. Consequently, $(A3)$ is valid for (m, o).

In proceeding to prove the validity of condition $(A4)$, let $X \in m$ be an arbitrary material. Then $X \in m^*$, and since (m^*, o^*) satisfies $(A4)$, there exists an operating unit $u = (\alpha, \beta) \in o^*$ such that $X \in \alpha \cup \beta$. This implies that there exists an operating unit $u_j = (\alpha_j, \beta_j) \in o$ such that $\varphi(u_j) = u$ and $X \in \alpha_j \cup \beta_j$. Indeed, in the opposite case, we would have that $X \notin \alpha \cup \beta$ which is a contradiction.

This completes the proof of Theorem 1.

Let us equip structural model $\mathbf{M}/\equiv$ with the weight function $\bar{w}$ defined as follows. For every $u \in O^*$, let $\bar{w}(u) = \sum_{u_t \in C(u')} w(u_t)$ where $\varphi(u') = u$. Since the equivalent operating units have an identical image, function $\bar{w}$ is well-defined. The constructed new model is then

$$\min\{\sum_{u \in o} \bar{w}(u) : (m, o) \in S(\mathbf{M}/\equiv)\}. \tag{2}$$

Extend the weight functions for the feasible solutions in the following way. For any $(m, o) \in S(\mathbf{M})$ and $(m^*, o^*) \in S(\mathbf{M}/\equiv)$, let $w(m, o) = \sum\{w(u) : u \in o\}$ and $\bar{w}(m^*, o^*) = \sum\{\bar{w}(u) : u \in o^*\}$. Then, $w(m, o) = \bar{w}(\varphi(m, o))$ is valid for all feasible solutions $(m, o) \in S(\mathbf{M})$. On the basis of this observation and Theorem 1, the validity of the following statement is obvious.

Theorem 2. *The image of an optimal solution of problem* (1) *under* φ *is an optimal solution of problem* (2)*, and conversely, the image of an optimal solution of problem* (2) *under* φ^{-1} *is an optimal solution of problem* (1).

To execute the merging reduction on an instance, we need to determine the equivalence relation introduced. For this reason, a further notation is introduced. Let $\mathbf{M} = (P, R, O)$ be a reduced structural model of PNS with $S(\mathbf{M}) \neq \emptyset$. Furthermore, let $u_j \in O$ be arbitrary. Then, we can construct a new structural model of PNS, $\mathbf{M}(u_j) = (P, R, O \setminus \{u_j\})$. Let us denote the maximal structure of $\mathbf{M}(u_j)$ by (M_j, O_j) provided that it exists. If it does not exist, then let $M_j = O_j = \emptyset$. Then, we have the following statement.

Theorem 3. *For every* $u_i, u_j \in O$, $u_i \equiv u_j$ *if and only if* $u_i \in O \setminus O_j$ *and* $u_j \in O \setminus O_i$ *are simultaneously valid.*

Proof. Let us suppose that $u_i \in O \setminus O_j$ and $u_j \in O \setminus O_i$ for some $u_i \neq u_j \in O$. Let us consider an arbitrary feasible solution (m, o). We have to distinguish three cases.

Case 1. (m, o) does not contain u_i. Then, (m, o) is a subset of (M_i, O_i), and thus, by our assumption, (m, o) does not contain u_j.

Case 2. (m, o) does not contain u_j. In this case, (m, o) is a subset of (M_j, O_j), and hence, by our assumption, (m, o) does not contain u_i.

Case 3. (m, o) contains both u_i and u_j.

Since there is no further case, we have proved that $u_i \equiv u_j$.

In order to prove the necessity of the condition, let us suppose that $u_i \equiv u_j$ for some $u_i \neq u_j \in O$. Let us consider the structural models $\mathbf{M}(u_i)$ and $\mathbf{M}(u_j)$. Then, (M_j, O_j) is the union of those feasible solutions which do not contain u_j provided that there exists such a feasible solution. Since $u_i \equiv u_j$ none of these feasible solutions contains u_i. Consequently, their union does not contain u_i, *i.e.*, $u_i \in O \setminus O_j$. We can obtain by a similar argument that $u_j \in O \setminus O_i$. If every feasible solution contains u_j, *i.e.*, $O_j = \emptyset$, then from $u_i \equiv u_j$, it follows that every feasible solution contains u_i as well, and thus $O_i = \emptyset$, and the corresponding inclusions are obviously valid.

From Theorem 3, we get immediately the following corollary.

Corollary. *If $O_j = O \setminus \{u_j\}$, then u_j is not mergeable with any other operating unit.*

Now, by Theorem 3 and the Maximal Structure Generation algorithm, we obtain the following procedure to determine the required equivalence relation where it is assumed that $O = \{u_1, \ldots, u_n\}$.

Procedure

Initialization

- *Step 1.* Set $i := 1$, $k := 1$, $N = \{1, \ldots, n\}$.
- *Step 2.* Determine the maximal structure of $\mathbf{M}(u_i)$ by the maximal structure generation algorithm. If $O_i = O \setminus \{u_i\}$, then let $V_k = \{u_i\}$, $N = N \setminus \{i\}$, $k = k + 1$. Proceed to Step 3.
- *Step 3.* If $i = n$, then proceed to Step 4. Otherwise, let $i = i + 1$, and proceed to Step 2.
- *Step 4.* Terminate if $N = \emptyset$. Otherwise, let i denote the smallest element of N. Let $J = \{t : t \in N \ \& \ u_t \in O \setminus O_i\}$. Let $V = \emptyset$, and proceed to Step 5.
- *Step 5.* If $J = \emptyset$, then let $N = N \setminus \{i\}$, $V_k = V \cup \{u_i\}$, $k = k + 1$, and proceed to Step 4. Otherwise, proceed to Step 6.
- *Step 6.* Choose an element j from J. Let $J = J \setminus \{j\}$. If $u_i \in O \setminus O_j$, then let $V = V \cup \{u_j\}$, $N = N \setminus \{j\}$. Proceed to Step 5.

As a result of this procedure, we obtain the equivalence classes belonging to the required equivalence relation as $V_1, \ldots, V_k$.

Regarding the merging reduction one can raise the following questions.

(1) *Does the merging reduction decrease the measure of practical problems or is it only a theoretical aspect?*

(2) *Is the decrease of the measure able to balance the higher complexity of the operating units caused by the merging reduction with respect to the running times of the known procedures for solving PNS problems?*

Both questions were investigated empirically. The corresponding computational experiences and their results are presented in the following section.

3 Empirical analysis

The first empirical analysis is devoted to the estimation of the decrease of measure. More precisely, it was investigated that how large the decrease of the model size was in general. For this reason, we considered 1000 randomly generated PNS problems (for their generation *cf.* [11]), and for each problem, the maximal structure was determined, then the merging reduction was performed. Figure 1 shows the average numbers of the operating units in the initial problem, in the maximal structure, and in the problem after the merging reduction. Figure 2 presents the same information in percent.

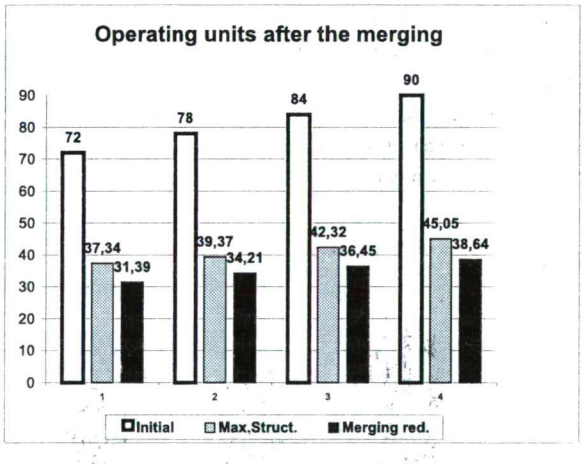


Figure 1: Average number of operating units.

As the results of the empirical analysis show, the merging reduction results in a decrease of 7% in general. It is obvious that the price of this decrease is that the new problem will be more complex than the initial one, namely, the operating units will have more input and output materials. Therefore, it is interesting to study the behaviours of the available procedures for solving PNS problems on the problems obtained by merging reduction. For this reason, we executed the following empirical investigation. Three procedures, the Accelerated Branch-and-Bound Algorithm, ABBA in short (see [8]), the Modified Accelerated Branch-and-Bound Procedure, in short MABBA (*cf.* [9]), and a version of the Refined Modified Accelerated Branch-and-Bound Procedure, in short RMABBA [11] were involved in

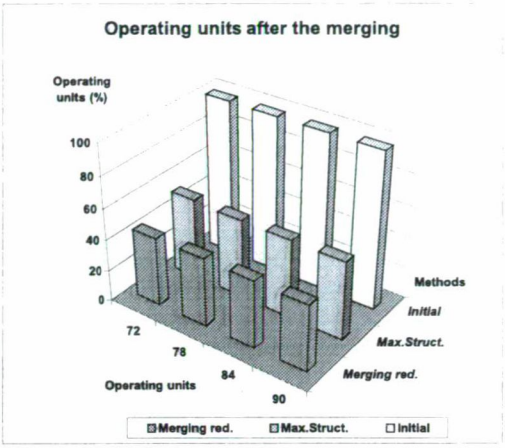


Figure 2: Average number of operating units in percent.

the empirical analysis. 1000 PNS problems with 100 materials were generated randomly, and for each of them the maximal structure was determined and the merging reduction was performed as well. Then, the two problems (problem belonging to the maximal structure and problem obtained by the merging reduction) were solved by the three procedures considered. Figure 3 shows the averages of the running times in percent for the different procedures.

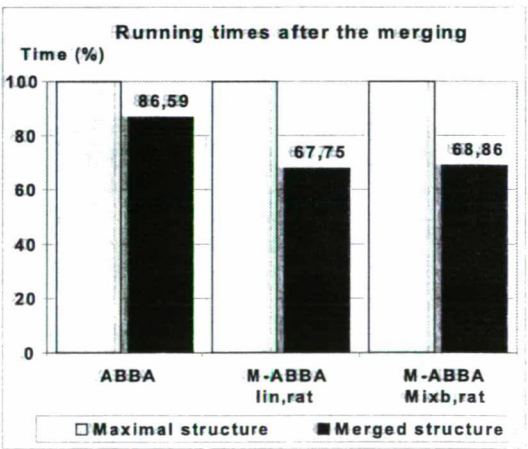


Figure 3: Behaviours of the procedures.

Conclusions. The empirical analysis shows that the merging reduction is appropriate to get further reduction of the model, moreover, the higher complexity of the operating units not necessarily implies longer running time for the procedures

considered. The smaller measure of the PNS problem resulted in a smaller running time for the procedures investigated even if the complexity of the operating units became higher.

References

[1] Blázsik, Z. and B. Imreh, A note on connection between PNS and set covering problems, *Acta Cybernetica* **12** (1996), 309-312.

[2] Fülöp, J., B. Imreh, F. Friedler, On the reformulation of some classes of PNS problems as set covering problems, *Acta Cybernetica*, **13** (1998), 329-397.

[3] Friedler, F., L. T. Fan, B. Imreh, Process Network Synthesis: Problem Definition, *Networks* **28** (1998), 119-124.

[4] Friedler, F., K. Tarján, Y. W. Huang, and L. T. Fan, Graph-Theoretic Approach to Process Synthesis: Axioms and Theorems, *Chem. Eng. Sci.* **47**(8) (1992), 1973-1988.

[5] Friedler, F., K. Tarján, Y.W. Huang, and L.T. Fan, Combinatorial Algorithms for Process Synthesis, *Computers chem. Engng.* **16** (1992), S313-S320.

[6] Friedler, F., K. Tarján, Y. W. Huang, and L. T. Fan, Graph-Theoretic Approach to Process Synthesis: Polinomyal Algorithm for maximal structure generation, *Computer chem. Engng.* **17** (1993), 924-942.

[7] Friedler, F., K. Tarján, Y. W. Huang, and L. T. Fan, Combinatorial Algorithms for Process Synthesis, *Computer chem. Engng.* **16** (1992), 313-320.

[8] Friedler, F., J. B. Varga, E. Fehér, and L. T. Fan, Combinatorially Accelerated Branch-and-Bound Method for Solving the MIP Model of Process Network Synthesis, Nonconvex Optimization and its Applications, Kluwer Academic Publisher, Norwell, MA, U.S.A. (in press).

[9] Imreh, B., F. Friedler, L. T. Fan, An Algorithm for Improving the Bounding Procedure in Solving Process Network Synthesis by a Branch-and-Bound Method *Developments in Global Optimization*, editors: I. M. Bonze, T. Csendes, R. Horst, P. M. Pardalos, Kluwer Academic Publisher, Dordrecht, Boston, London, 1996, 301-348.

[10] Imreh, B., J. Fülöp, F. Friedler, On the Equivalence of the Set Covering and Process Network Synthesis Problems, *Networks*, submitted for publication.

[11] Imreh, B., G. Magyar, Empirical Analysis of Some Procedures for Solving Process Network Synthesis Problem, *Journal of Computing and Information Technology*, to appear.

Acta Cybernetica 14 (1999) 263-283.

Construction of Recursive Algorithms for Polarity Matrices Calculation in Polynomial Logical Function Representation

Dragan Janković *

Abstract

There is no algorithm for the calculation of optimal fixed polarity expansion. Therefore, the efficient calculation of polarity matrix consisting of all fixed polarity expansion coefficients is very important task. We show that polarity matrix can be generated as convolution of function f with rows of relates transform matrix. The recursive properties of the convolution matrix affect to properties of polarity matrix. In literature are known some recursive algorithms for the calculation of polarity matrix of some expressions for Multiple-valued (MV) functions [3,6]. We give a unique method to construct recursive procedures for the polarity matrices calculation for any Kronecker product based expression of MV functions. As a particular cases we derive two recursive algorithms for calculation of fixed polarity Reed-Muller-Fourier expressions for four-valued functions.

1 Introduction

Compact representation of switching functions is not only the mater of notation convenience, but highly relates to the analysis and synthesis of these functions. Both analysis and synthesis procedures, as well as final realizations, can be greatly simplified by choosing appropriate representations of switching functions.

In the case of Reed-Muller (RM) expressions, the problem to determine the most compact representation reduces to the determination of optimal polarity for switching variables. By choosing between the positive or negative literals for each variable, but not both at the same time, the Fixed polarity RM (FPRM) expressions are defined [5].

In a FPRM, the number of products, or equivalently, the number of non-zero coefficients may be considerably reduced by choosing different polarities for the variables. The FPRM with the minimum number of products is taken as the

*Faculty of Electronic Engineering, University of Niš, Yugoslavia

optimal FPRM for f. If there are two FPRMs with the same number of products, the one with the smaller number of literals in the products is taken.

There is no method to determine apriori the polarities of variables for a given function f. In practice, it is necessary to generate all the FPRMs and chose the optimal one. That can be efficiently done by generating the polarity matrices $\mathbf{P}_{RM}$ whose rows are RM-coefficients for the given f with different polarities of variables. The efficiency of generation of $\mathbf{P}_{RM}$ is based upon its recursive structure originating in the Kronecker product representation of the RM-transform matrix.

Polynomial representations of Multiple-valued (MV) functions are very interesting with advent of multiple-valued circuit technology, in particular recent experience with current-mode circuits that are very attractive for implementation of MV functions. Specially, the realization of the corresponding 4-valued circuit is very efficient. The problem of compact representations is even harder in the case of MV functions. Galois field (GF) expressions are a generalization of RM-expressions to MV case [7]. Optimization of GF-expressions can be studied and solved in a way similar to that used for RM-expressions. In particular, efficient methods for generation of polarity matrices $\mathbf{P}_{GF}$ for GF-expressions of ternary functions are reported in [6], while the corresponding methods for quaternary functions are reported in [3], and further elaborated in [1], [2], [4].

Reed-Muller-Fourier (RMF) expressions are an alternative extension of RM-expressions to MV case [8]. It has been shown that RMF-expressions require on the average smaller number of products than GF-expressions to represent a given function f [9]. The optimization of RMF-expressions is performed in the same way as in the GF-expressions by choosing different polarities for the variables. As in the case of RM and GF-expressions, there are no methods to determine apriori the polarity for the variables in a given f to get the RMF-expression with the minimum number of products. For that reason, the efficient calculation of polarity matrices is a very important task. An analyse of present recursive methods for calculation of polarity matrix for some particular expressions shows that recursive approaches are more efficient than others methods. Therefore, the construction of recursive relations for polarity matrix calculation for various expressions are a very interesting problem.

In this paper, we uniformly consider the coefficients in various expressions for logic functions as spectral coefficients in particular spectral transforms. We show that polarity matrix can be generated as convolution of f with columns of related transform matrix. The recursive properties of the polarity matrix result from properties of the convolution matrix. We give a unique method to construct recursive procedures for the polarity matrices calculation for any Kronecker product based expression of MV functions.

This method involves existing methods as a particular cases and permits various generalizations. For illustration, we derive two recursive algorithms for calculation of fixed polarity Reed-Muller-Fourier expressions for four-valued functions.

2 Notations and Definitions

Definition 1 *Let* $\mathbf{E}(q)$ *be the set of integers modulo* q. *n-variable q-valued logical function is mapping*

$$f : \mathbf{E}(q)^n \to \mathbf{E}(q).$$

Definition 2 *Each n-variable q-valued logical function f can be represented in polynomial form*

$$\begin{aligned} f(x_1, \ldots, x_n) &= c_0 \oplus c_1 x_n \oplus c_2 x_n^2 \oplus \ldots \oplus c_{q-1} x_n^{q-1} \oplus c_q x_{n-1} \\ &\quad \oplus c_{q+1} x_n x_{n-1} \oplus \ldots \oplus c_{q^n-1} x_n x_{n-1} \ldots x_1. \end{aligned}$$

The coefficient vector $\mathbf{C}$, consisting from the coefficients $c_i, i = 0, \ldots, q^n - 1$ can be calculated as direct transform of function f, given by its truth vector $\mathbf{F} = [f(0), \ldots, f(q^n - 1)]^T$ i.e.

$$\begin{aligned} \mathbf{C} = (c_0, c_1, \ldots, c_{q^n-1}) &= \mathbf{T}_n \mathbf{F} = \left(\bigotimes_{i=1}^{n} \mathbf{T}_1 \right) \cdot \mathbf{F} \\ &= \left(\bigotimes_{i=1}^{n} \begin{bmatrix} 1 & x_i & x_i^2 & \ldots & x_i^{q-1} \end{bmatrix}^{-1} \right) \cdot \mathbf{F}, \end{aligned} \quad (1)$$

where with $^{-1}$ is denoted the inverse matrix and $\otimes$ denotes Kronecker product. $\mathbf{T}_n$ is transform matrix.

The number of non-zero coefficients in vector $\mathbf{C}$ is usually used criteria of optimality. Optimization can be made by using different polarities of variables.

Definition 3 *i-th polarity of variable x in notation $\overset{i-}{x}$ is defined as: $\overset{i-}{x} = x \oplus i, i = 0, ..., q-1$, for q-valued functions.*

If each literal x_i in expansion (1) have complemented or noncomplemented form but not both this expansion is named fixed polarity expansion. For n-variable q-valued function the number of different polarities is q^n.

Theorem 1 *For polarity $k = (k_1, \ldots, k_n)$ ($< k >= \sum_{i=1}^{n} k_i q^{n-i}$), the coefficient vector can be calculated as [6]:*

$$\mathbf{C}^{<k>} = \mathbf{T}_n^{<k>} \cdot \mathbf{F} = \left(\bigotimes_{i=1}^{n} \mathbf{T}_1^{(k_i)} \right) \cdot \mathbf{F}, \quad (2)$$

where $\mathbf{T}_1^{(k_i)}$ is the matrix $\mathbf{T}_1$ whose that columns are shifted for k_i places in according to the definition of operation $\oplus$.

Example 1 *Let f is two variable function on Galois field $GF(3)$. The operations $\cdot$ and $\oplus$ are multiplication modulo 3 and addition modulo 3 respectively. $< k >$ polarity expansion coefficient vector of f is given as:*

$$\mathbf{C}^{<k>} = \left(\mathbf{T}_1^{(k_1)} \otimes \mathbf{T}_1^{(k_2)} \right) \cdot \mathbf{F}.$$

If $< k >=< 7 >$ *then* $c^{<7>}$ *is calculated as:*

$$\mathbf{T}_1^{-1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 2 & 1 \end{bmatrix}, \mathbf{T}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix},$$

$$\mathbf{T}_1^{(1)} = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{bmatrix}, \mathbf{T}_1^{(2)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 2 & 2 & 2 \end{bmatrix}.$$

$$\begin{aligned} c^{<7>} &= \left(\mathbf{T}_1^{(2)} \otimes \mathbf{T}_1^{(1)}\right) \cdot \mathbf{F} \\ &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 2 & 2 & 2 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 1 \\ 2 & 1 & 0 \\ 2 & 2 & 2 \end{bmatrix} \cdot \mathbf{F} \\ &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 \\ 2 & 1 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \\ 2 & 2 & 2 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 \\ 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \mathbf{F}. \end{aligned}$$

Lemma 1 *The coefficient vector of polarity* $< p >$ *can be calculated as*

$$\begin{aligned} \mathbf{C}^{<p>} &= \mathbf{T}_n^{<p>}\mathbf{F} = \left(\bigotimes_{i=1}^{n} \mathbf{T}_1^{<p>}\right) \cdot \mathbf{F} = \mathbf{T}_n \cdot \mathbf{F}^{<p>} = \mathbf{T}_n \cdot \mathbf{F}^{<p_1,p_2,\ldots,p_n>} \\ &= \mathbf{T}_n \cdot \mathbf{F}\left(x_1 \oplus p_2, x_2 \oplus p_2, \ldots, x_n \oplus p_n\right). \end{aligned} \tag{3}$$

Example 2 *Let* f *is the two variable* 3*-valued function defined on* $GF(3)$ *and represented by truth vector* $\mathbf{F} = (122010210)$. *The vector* $c^{<7>}$ *can be calculated as*

$$c^{<7>} = \mathbf{T}_1^{(2)} \otimes \mathbf{T}_1^{(1)} \cdot \mathbf{F}^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 \\ 2 & 1 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \\ 2 & 2 & 2 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 \\ 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 2 \\ 0 \\ 1 \\ 0 \\ 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 2 \\ 2 \\ 1 \\ 1 \\ 2 \\ 0 \end{bmatrix},$$

$$c^{<7>} = \mathbf{T}_2 \cdot \mathbf{F}^{<7>} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 2 \\ 1 \\ 2 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 2 \\ 2 \\ 1 \\ 1 \\ 2 \\ 0 \end{bmatrix}.$$

Definition 4 *The polarity matrix* $\mathbf{P}$ *of an* n*-variable* q*-valued function* $f(x_1, x_2, \ldots, x_n)$ *is a* $(q^n \times q^n)$ *matrix where every row matches a coefficient vector in a different polarity* $< k >$*.* i*-th row corresponds to a coefficient vector in the* $< i >$*-th polarity, i.e.,* $c^{<i>}$*.*

Definition 5 *The optimal polarity of function* $f(x_1, x_2, \ldots, x_n)$ *is defined as polarity* k_{opt} *whose coefficient vector has the minimal number of nonzero elements.*

Example 3 *The polarity matrix of a two variable quaternary function* f*, given by truth vector* $\mathbf{F} = (0311132322321002)$ *is given as*

$$\mathbf{P} = \begin{bmatrix} 0 & 3 & 1 & 3 & 1 & 1 & 2 & 2 & 0 & 3 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 2 & 2 & 1 & 0 & 3 & 0 & 2 & 1 & 1 & 0 & 2 & 0 & 0 & 0 & 2 \\ 1 & 0 & 1 & 1 & 1 & 3 & 2 & 2 & 0 & 1 & 2 & 2 & 0 & 0 & 2 & 2 \\ 1 & 3 & 0 & 3 & 2 & 1 & 0 & 2 & 3 & 3 & 0 & 2 & 0 & 2 & 0 & 2 \\ 1 & 2 & 3 & 1 & 1 & 2 & 0 & 0 & 2 & 1 & 0 & 0 & 2 & 0 & 2 & 2 \\ 3 & 3 & 2 & 3 & 3 & 2 & 0 & 0 & 1 & 1 & 0 & 0 & 2 & 2 & 0 & 2 \\ 2 & 1 & 3 & 3 & 1 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 2 & 2 \\ 3 & 2 & 0 & 1 & 3 & 2 & 0 & 0 & 3 & 1 & 0 & 0 & 2 & 0 & 0 & 2 \\ 2 & 0 & 3 & 1 & 3 & 1 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 2 & 3 & 2 & 1 & 0 & 0 & 3 & 3 & 0 & 2 & 0 & 0 & 0 & 2 \\ 3 & 3 & 3 & 3 & 1 & 1 & 0 & 0 & 0 & 3 & 2 & 2 & 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 2 & 0 & 2 & 0 & 2 \\ 1 & 3 & 3 & 1 & 3 & 0 & 2 & 2 & 2 & 3 & 0 & 0 & 2 & 0 & 2 & 2 \\ 0 & 0 & 2 & 3 & 3 & 2 & 0 & 2 & 3 & 3 & 0 & 0 & 2 & 2 & 0 & 2 \\ 0 & 2 & 3 & 3 & 1 & 2 & 2 & 2 & 0 & 3 & 0 & 0 & 0 & 2 & 2 & 2 \\ 2 & 3 & 0 & 1 & 3 & 0 & 0 & 2 & 1 & 3 & 0 & 0 & 2 & 0 & 0 & 2 \end{bmatrix} \begin{matrix} c^{<0>} \\ c^{<1>} \\ c^{<2>} \\ c^{<3>} \\ c^{<4>} \\ c^{<5>} \\ c^{<6>} \\ c^{<7>} \\ c^{<8>} \\ c^{<9>} \\ c^{<10>} \\ c^{<11>} \\ c^{<12>} \\ c^{<13>} \\ c^{<14>} \\ c^{<15>} \end{matrix}$$

3 Convolution

Definition 6 *Convolution of* n*-variable* q*-valued logic functions* f *and* g *is defined as*

$$f \star g(s) = \sum_{x=0}^{q^n-1} f(x) \cdot g(x \oplus s),\ s = 0, 1, \cdots, q^n - 1,$$

$$
\begin{aligned}
f \star g(s_1, \cdots, s_n) &= \sum_{x=(0,0,\cdots,0)}^{q-1,q-1,\cdots,q-1} f(x_1, \cdots, x_n) \cdot g(x_1 \oplus s_1, \cdots, x_n \oplus s_n), \\
s &= (s_1, s_2, \cdots, s_n), \quad s = \sum_{i=1}^{n} s_i \cdot q^{n-i}, \\
x &= (x_1, x_2, \cdots, x_n), \quad x = \sum_{i=1}^{n} x_i \cdot q^{n-i}.
\end{aligned}
$$

Operations $\oplus$ and $\cdot$ are defined on corresponding algebraic structure.

The convolution matrix is given as:

$$
\mathbf{G}_{conv} = \begin{bmatrix}
g(0 \oplus 0) & g(1 \oplus 0) & \cdots & g((q^n-1) \oplus 0) \\
g(0 \oplus 1) & g(1 \oplus 1) & \cdots & g((q^n-1) \oplus 1) \\
\vdots & \cdots & \cdots & \vdots \\
g(0 \oplus (q^n-1)) & g(1 \oplus (q^n-1)) & \cdots & g((q^n-1) \oplus (q^n-1))
\end{bmatrix}. \tag{4}
$$

Now, the convolution of f and g, in according to (4) can be write in form

$$
f \star g = \mathbf{G}_{conv} \cdot f. \tag{5}
$$

Theorem 2 *Convolution of k-th row in transform matrix t_k, with function vector $\mathbf{F}$ gives the vector of k-th coefficients in polarity matrix i.e. $\mathbf{P}^k = t_k \star \mathbf{F}$.*

The proof of theorem can be done from the structure of convolution matrix.
Proof:

$$
\begin{aligned}
\mathbf{P}_n &= \begin{bmatrix} \mathbf{C}^{(0)} \\ \mathbf{C}^{(1)} \\ \vdots \\ \mathbf{C}^{(q^n-1)} \end{bmatrix} = \begin{bmatrix} \mathbf{T}_n \cdot \mathbf{F}^{(0)} \\ \mathbf{T}_n \cdot \mathbf{F}^{(1)} \\ \vdots \\ \mathbf{T}_n \cdot \mathbf{F}^{(q^n-1)} \end{bmatrix} \\
&= \begin{bmatrix}
\sum_{j=0}^{q^n-1} \mathbf{T}_n(0,j)\mathbf{F}(j \oplus 0) & \cdots & \sum_{j=0}^{q^n-1} \mathbf{T}_n(q^n-1,j)\mathbf{F}(j \oplus 0) \\
\sum_{j=0}^{q^n-1} \mathbf{T}_n(0,j)\mathbf{F}(j \oplus 1) & \cdots & \sum_{j=0}^{q^n-1} \mathbf{T}_n(q^n-1,j)\mathbf{F}(j \oplus 1) \\
\vdots & \cdots & \vdots \\
\sum_{j=0}^{q^n-1} \mathbf{T}_n(0,j)\mathbf{F}(j \oplus q^n-1) & \cdots & \sum_{j=0}^{q^n-1} \mathbf{T}_n(q^n-1,j)\mathbf{F}(j \oplus q^n-1)
\end{bmatrix} \\
&= \begin{bmatrix} \mathbf{P}^0 & \mathbf{P}^1 & \cdots & \mathbf{P}^{q^n-1} \end{bmatrix},
\end{aligned}
$$

where

$$
\begin{aligned}
\mathbf{P}^k &= \left[\mathbf{P}^k(0), \mathbf{P}^k(1), \cdots, \mathbf{P}^k(q^n-1)\right], \; k = 0, \cdots, q^n-1, \\
\mathbf{P}^k(i) &= \sum_{j=0}^{q^n-1} \mathbf{T}_n(k,j)\mathbf{F}(j \oplus i), \; i = 0, \cdots, q^n-1.
\end{aligned}
$$

It follows from equation (4)

$$\mathbf{P}^k = \mathbf{T}_n^k \star \mathbf{F}.$$

Example 4 *Tenth column from polarity matrix for function f from example* 3 *can be calculated as convolution of tenth row in transform matrix and truth-vector* $\mathbf{F}$*:*

$$\begin{bmatrix} 3 \\ 1 \\ 0 \\ 0 \\ 2 \\ 2 \\ 0 \\ 0 \\ 3 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \star \begin{bmatrix} 0 \\ 3 \\ 1 \\ 1 \\ 1 \\ 3 \\ 2 \\ 3 \\ 2 \\ 2 \\ 3 \\ 2 \\ 2 \\ 0 \\ 0 \\ 2 \end{bmatrix} = \mathbf{A} + \mathbf{B} = \begin{bmatrix} 3 \\ 1 \\ 1 \\ 3 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 3 \\ 3 \\ 1 \\ 3 \\ 3 \\ 3 \\ 3 \end{bmatrix},$$

where

$$\mathbf{A} = \begin{bmatrix} 3\cdot 0+1\cdot 3+0\cdot 1+0\cdot 1+2\cdot 1+2\cdot 3+0\cdot 2+0\cdot 3 \\ 3\cdot 3+1\cdot 1+0\cdot 1+0\cdot 0+2\cdot 3+2\cdot 2+0\cdot 3+0\cdot 1 \\ 3\cdot 1+1\cdot 1+0\cdot 0+0\cdot 3+2\cdot 2+2\cdot 3+0\cdot 1+0\cdot 3 \\ 3\cdot 1+1\cdot 0+0\cdot 3+0\cdot 1+2\cdot 3+2\cdot 1+0\cdot 3+0\cdot 2 \\ 3\cdot 1+1\cdot 3+0\cdot 2+0\cdot 3+2\cdot 2+2\cdot 2+0\cdot 3+0\cdot 2 \\ 3\cdot 3+1\cdot 2+0\cdot 3+0\cdot 1+2\cdot 2+2\cdot 3+0\cdot 2+0\cdot 2 \\ 3\cdot 2+1\cdot 3+0\cdot 1+0\cdot 3+2\cdot 3+2\cdot 2+0\cdot 2+0\cdot 2 \\ 3\cdot 3+1\cdot 1+0\cdot 3+0\cdot 2+2\cdot 2+2\cdot 2+0\cdot 2+0\cdot 3 \\ 3\cdot 2+1\cdot 2+0\cdot 3+0\cdot 2+2\cdot 1+2\cdot 0+0\cdot 0+0\cdot 2 \\ 3\cdot 2+1\cdot 3+0\cdot 2+0\cdot 2+2\cdot 0+2\cdot 0+0\cdot 2+0\cdot 1 \\ 3\cdot 3+1\cdot 2+0\cdot 2+0\cdot 2+2\cdot 0+2\cdot 2+0\cdot 1+0\cdot 0 \\ 3\cdot 2+1\cdot 2+0\cdot 2+0\cdot 3+2\cdot 2+2\cdot 1+0\cdot 0+0\cdot 0 \\ 3\cdot 1+1\cdot 0+0\cdot 0+0\cdot 2+2\cdot 0+2\cdot 3+0\cdot 1+0\cdot 1 \\ 3\cdot 0+1\cdot 0+0\cdot 2+0\cdot 1+2\cdot 3+2\cdot 1+0\cdot 1+0\cdot 0 \\ 3\cdot 0+1\cdot 2+0\cdot 1+0\cdot 0+2\cdot 1+2\cdot 1+0\cdot 0+0\cdot 3 \\ 3\cdot 2+1\cdot 1+0\cdot 0+0\cdot 0+2\cdot 1+2\cdot 0+0\cdot 3+0\cdot 1 \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix}
3\cdot 2+1\cdot 2+0\cdot 3+0\cdot 2+0\cdot 1+0\cdot 0+0\cdot 0+0\cdot 2\\
3\cdot 2+1\cdot 3+0\cdot 2+0\cdot 2+0\cdot 0+0\cdot 0+0\cdot 2+0\cdot 1\\
3\cdot 3+1\cdot 2+0\cdot 2+0\cdot 2+0\cdot 0+0\cdot 2+0\cdot 1+0\cdot 0\\
3\cdot 2+1\cdot 2+0\cdot 2+0\cdot 3+0\cdot 2+0\cdot 1+0\cdot 0+0\cdot 0\\
3\cdot 1+1\cdot 0+0\cdot 0+0\cdot 2+0\cdot 1+0\cdot 0+0\cdot 3+0\cdot 1\\
3\cdot 0+1\cdot 0+0\cdot 2+0\cdot 1+0\cdot 0+0\cdot 3+0\cdot 1+0\cdot 0\\
3\cdot 0+1\cdot 2+0\cdot 1+0\cdot 0+0\cdot 0+0\cdot 1+0\cdot 1+0\cdot 3\\
3\cdot 2+1\cdot 1+0\cdot 0+0\cdot 0+0\cdot 2+0\cdot 1+0\cdot 0+0\cdot 1\\
3\cdot 0+1\cdot 3+0\cdot 1+0\cdot 1+0\cdot 0+0\cdot 1+0\cdot 3+0\cdot 3\\
3\cdot 3+1\cdot 1+0\cdot 1+0\cdot 0+0\cdot 3+0\cdot 3+0\cdot 2+0\cdot 1\\
3\cdot 1+1\cdot 1+0\cdot 0+0\cdot 3+0\cdot 1+0\cdot 2+0\cdot 3+0\cdot 3\\
3\cdot 1+1\cdot 0+0\cdot 3+0\cdot 1+0\cdot 1+0\cdot 3+0\cdot 1+0\cdot 2\\
3\cdot 1+1\cdot 3+0\cdot 2+0\cdot 3+0\cdot 2+0\cdot 2+0\cdot 3+0\cdot 2\\
3\cdot 3+1\cdot 2+0\cdot 3+0\cdot 1+0\cdot 2+0\cdot 3+0\cdot 2+0\cdot 2\\
3\cdot 2+1\cdot 3+0\cdot 1+0\cdot 3+0\cdot 3+0\cdot 2+0\cdot 2+0\cdot 2\\
3\cdot 3+1\cdot 1+0\cdot 0+0\cdot 2+0\cdot 2+0\cdot 2+0\cdot 2+0\cdot 3
\end{bmatrix}.$$

4 Calculation of the Polarity Matrix

The polarity matrix can be calculated directly with equation (2). The complexity of this direct method is $(q^n)^3$ i.e. practically unuseful for large q. For the calculation of polarity matrix can be used the FFT-like method. The complexity of this method is $n(q^n)^2$. In [3,6] is shown that the polarity matrices can be generated efficiently by recursive relations. Proposed procedure is given only for GF(3) "recursion by column" and "recursion by row" for GF(4). In this section, we give the unique method for the generation of recursive relations for the calculation of polarity matrix for arbitrary finite fields. Both "recursion by columns" and "recursion by rows" are considered. Our method is generalization of methods proposed in [3] and [6].

4.1 Unique method for the generation of recursive relations for the polarity matrix construction

If we have in mind that transform matrix is given in Kronecker product form, the next theorem is obviously.

Theorem 3 *Let* $\mathbf{T}(n)$ *is transformation matrix given as:* $\mathbf{T}(n) = \bigotimes_{i=1}^{n} \mathbf{T}_i$ *where the dimension of matrix* $\mathbf{T}_i$ *is* $q_i \times q_i$ *. Element from* p *row and* r *column* $C_{<r>}^{<p>}$ *in polarity matrix* $\mathbf{P}$ *is given as:*

$$C_{<r>}^{<p>} = C_{<r_1,r_2,\cdots,r_i,\cdots,r_n>}^{<p_1,p_2,\cdots,p_i,\cdots,p_n>} = \sum_{l=0}^{q_i-1} \mathbf{T}_i(r_i,l)\cdot C_{<r_1,r_2,\cdots,0,\cdots,r_n>}^{<p_1,p_2,\cdots,l,\cdots,p_n>},\ i=1,\cdots,n$$

If $\mathbf{T}_i = \mathbf{T}_j,\ \forall i,j \in \{1,2,\cdots,n\}$ *then*

$$C^{<p>}_{<r>} = C^{<p_1,p_2,\cdots,p_i,\cdots,p_n>}_{<r_1,r_2,\cdots,r_i,\cdots,r_n>} = \sum_{l=0}^{q_i-1} \mathbf{T}(r_i,l) \cdot C^{<p_1,p_2,\cdots,l,\cdots,p_n>}_{<r_1,r_2,\cdots,0,\cdots,r_n>},\ i = 1,\cdots,n$$

or in matrix form

$$\mathbf{P} = P_{q^n \times q^n(n)} = \left[p^{n-1}(i,j)\right],\ p^{n-1}(i,j) = \sum_{l=0}^{q-1} T(j,l) \cdot p^{n-1}(l \oplus i, 0),\ j \neq 0. \quad (6)$$

This relation is recurrence by columns. ¿From recurrence relation (6) by first columns where we start from the first column i.e. 0-column it can be derived recurrence matrix relation started from any column k. Derived recurrence matrix relation we called "recurrence by k-th column". Recurrence by k-th column can be derived if we each element in k-th column from (6) denote with one letter and calculate relations. In this manner can be calculated "recurrence by k-th row", too.

The formal method for construction recurrence matrix relation for polarity matrix calculation may be presented through following steps:

1. The generation of $q \times q$ simbolic matrix $\mathbf{B}$ as $\mathbf{B} = \left[\mathbf{B}^{i \oplus j}\right],\ 0 \leq i,j \leq q-1$.
2. The generation of $q \times q$ matrix $\mathbf{Q} = \mathbf{T}^{-1} \cdot \mathbf{B}$.
3. If it wish the recurrence by k-th column/row, the elements from k-th column/row are substituted with one letter $P^i,\ 0 \leq i \leq q-1$. These substitutions give equations system consisting of q equations.
4. Solving the generating equations system.
5. The modification of the matrix $\mathbf{Q}$ in according to the solutions of previous equation system.
6. The substitution $\mathbf{Q}$ with P_n and P^i with P^i_{n-1} .

This method can be generalized to the case when matrices $\mathbf{T}_i$ are different, e.i. $\mathbf{T}_n = \bigotimes_{i=1}^{n} \mathbf{T}_i,\ \mathbf{T}_i \neq \mathbf{T}_j$ if $i \neq j$. In this case, it is not possible to generate polarity matrix by only one recurrence matrix equation. The polarity matrix can be generated by means n recurrence matrix equations similar to the above matrix equations. For each of n steps, we generate recurrence matrix relations based on the matrix $\mathbf{T}_i$. Namely, we run above method n times, substituting $\mathbf{T}_1^{-1}$ with $\mathbf{T}_i^{-1},\ 1 \leq i \leq n$. Obviously, dimensions of matrices $\mathbf{B}$ and $\mathbf{Q}$ are equal $q_i \times q_i$. This will be illustrated in following example.

Example 5 *Let* $\mathbf{T} = \mathbf{T}_1 \otimes \mathbf{T}_2$, $\mathbf{T}_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, $\mathbf{T}_1^{-1} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$,

$$\mathbf{T}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 3 & 2 & 1 \end{bmatrix},\ \mathbf{T}_2^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 3 & 2 \\ 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

The recurrence matrix equations are:

$$P_n^1 = \begin{bmatrix} P_{n-1}^0 & P_{n-1}^0 + P_{n-1}^1 \\ P_{n-1}^1 & P_{n-1}^0 + P_{n-1}^1 \end{bmatrix},$$

$$P_n^2 = \begin{bmatrix} P_{n-1}^0 & P_{n-1}^1 + 3P_{n-1}^2 + 2P_{n-1}^3 & P_{n-1}^1 + 2P_{n-1}^2 + 3P_{n-1}^3 & S \\ P_{n-1}^1 & P_{n-1}^2 + 3P_{n-1}^3 + 2P_{n-1}^0 & P_{n-1}^2 + 2P_{n-1}^3 + 3P_{n-1}^0 & S \\ P_{n-1}^2 & P_{n-1}^3 + 3P_{n-1}^0 + 2P_{n-1}^1 & P_{n-1}^3 + 2P_{n-1}^0 + 3P_{n-1}^1 & S \\ P_{n-1}^3 & P_{n-1}^0 + 3P_{n-1}^1 + 2P_{n-1}^2 & P_{n-1}^0 + 2P_{n-1}^1 + 3P_{n-1}^2 & S \end{bmatrix},$$

where

$$S = P_{n-1}^0 + P_{n-1}^2 + P_{n-1}^2 + P_{n-1}^3.$$

Proposed above method we explain in next section for the case of polarity matrix of Reed-Muller-Fourier expression of quaternary functions.

5 RMF-expressions for Quaternary Functions

To make the paper self-contained, we present in this section basic definitions for RMF-expressions for quaternary functions. Then, we consider their optimization by choosing different polarities for variables. It is assumed single polarity for a variable in the expression. In that way the Fixed polarity RMF (FPRMF) expressions are defined.

Let $E(4)$ be the set of integers modulo 4 with the addition and multiplication modulo 4 shown in Table 1 and Table 2.

Table 1: Addition modulo 4.

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Table 2: Multiplication modulo 4.

$\cdot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Define the exponentiation 4EXP and multiplication 4AND, denoted by $*$ and $\circ$, respectively, as in Table 3 and Table 4. Denote by $\mathbf{J}$ the space of n-variable quaternary functions, i.e., $f : \mathbf{E}(4)^n \to \mathbf{E}(4)$.

Definition 7 *The operator $D(n)$ in $\mathbf{J}$ is defined, in the matrix notation, by a $(4^n \times 4^n)$ diagonal matrix given by $\mathbf{D}(n) = diag(3, 1, \cdots, 1)$.*

Definition 8 *RMF-expression of a function $f \in \mathbf{J}$ given by its truth-vector $\mathbf{F} = [f(0), \ldots, f(4^n \times 4^n)]^T$ is given by [9]*

Table 3: Exponentiation 4EXP.

$*$	0	1	2	3
0	3	0	0	0
1	3	1	0	0
2	3	2	3	0
3	3	3	1	1

Table 4: Multiplication 4AND.

$\circ$	0	1	2	3
0	0	0	0	0
1	0	3	2	1
2	0	2	0	2
3	0	1	2	3

$$f(x_1,\cdots,x_n) = \left(\mathbf{D}(n)\left(\bigotimes_{i=1}^{n}[\ 1 \quad x_i \quad x_i^{*2} \quad x_i^{*3}\ ]\right)\right)\cdot\mathbf{A}, \tag{7}$$

where $\mathbf{A} = [a(0),\ldots,a(4^n-1)]^T$ *is the vector of RMF-coefficients determined by the matrix relation*

$$\mathbf{A} = \mathbf{R}(n)\cdot\mathbf{F},$$

where

$$\mathbf{R}(n) = 3\bigotimes_{i=1}^{n}\mathbf{R}_i,\quad \mathbf{R}_i = \begin{bmatrix} 1 & 0 & 0 & 0\\ 1 & 3 & 0 & 0\\ 1 & 2 & 1 & 0\\ 1 & 1 & 3 & 3 \end{bmatrix}.$$

In this relation, $\otimes$ denotes the Kronecker product and x_i^{*j}, $j\in\{2,3\}$ denotes the j-th power of x_j with respect to 4EXP.

In (7), the addition and multiplication are performed modulo 4.

6 Fixed Polarity RMF-expressions

Similarly as for RM-expressions for switching functions, and GF-expressions for MV functions, optimisation of RMF-expressions means reduction of the number of products, i.e., the number of non-zero RMF-coefficients. As noted above, the optimisation of RMF-expressions is possible if we use different polarities for the variables. For a p-valued variable, we consider $p-1$ complements defined by $\overset{i-}{x} = x\oplus i$, $i\in\{1,\cdots,p-1\}$. Thus, in a FPRMF-expression, a variable can appear as the positive literal x_i or any of $p-1$ negative literals $\overset{i-}{x}$, but not as few of them at the same time. Therefore, there are p^n different polarity FPRM-expressions for a given n-variable function f. For $p=4$, the complements are $\overset{1-}{x}$, $\overset{2-}{x}$, $\overset{3-}{x}$, and thus, there exist 4^n different FPRMF-expressions for a quaternary function f. These

different possible FPRMF-expressions are determined through the polarity vector $\mathbf{H} = (h_1, \cdots, h_n)$, where the value of $h_i \in \{0, 1, 2, 3\}$ determines polarity of the literal chosen for the variable x_i.

Definition 9 *For $f \in \mathbf{J}$ given by the truth-vector $\mathbf{F}$, the FPRMF-expression with the polarity vector $\mathbf{H} = (h_1, \cdots, h_n)$ is given by*

$$f(x_1, \cdots, x_n) = \left(\mathbf{D}(n)\left(\bigotimes_{i=1}^{n}\left[\begin{array}{cccc} 1 & {}^{h_i-}x_i & {}^{h_i-*2}x_i & {}^{h_i-*3}x_i \end{array}\right]\right)\right)\left(\left(3\bigotimes_{i=1}^{n}\mathbf{R}_i^{h_i}\right)\mathbf{F}\right), \quad (8)$$

where $\mathbf{R}_i^{h_i}$ is derived from $\mathbf{R}_i$ by the cyclic shift of its columns for h_i places. Thus,

$$\mathbf{R}_1^0 = \mathbf{R}_1 = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 1 & 2 & 3 \end{array}\right], \mathbf{R}_1^1 = \left[\begin{array}{cccc} 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 \\ 1 & 3 & 3 & 1 \end{array}\right],$$

$$\mathbf{R}_1^2 = \left[\begin{array}{cccc} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 3 \\ 1 & 0 & 1 & 2 \\ 3 & 3 & 1 & 1 \end{array}\right], \mathbf{R}_1^3 = \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 1 & 3 & 0 \\ 0 & 1 & 2 & 1 \\ 3 & 1 & 1 & 3 \end{array}\right].$$

Example 6 *The zero-polarity FPRMF-expression ($\mathbf{H} = [0, 0]$) for two-variable function f, given by the truth vector $\mathbf{F} = [0311132322321002]^T$ is*

$$\begin{aligned} f = {} & 3x_2 \oplus x_2^{*2} \oplus 3x_2^{*3} \oplus x_1 \oplus x_1 \circ x_2 \oplus 2x_1 \circ x_2^{*2} \oplus 2x_1 \circ x_2^{*3} \oplus 3x_1^{*2} \circ x_2 \otimes \\ & \otimes 2x_1^{*2} \circ x_2^{*2} \oplus 2x_1^{*2} \circ x_2^{*3} \oplus 2x_1^{*3} \oplus 2x_1^{*3} \circ x_2 \oplus 2x_1^{*3} \circ x_2^{*3} \oplus 2x_1^{*3} \circ x_2^{*3}. \end{aligned}$$

Definition 10 *For a given $f \in \mathbf{J}$, the FPRMF-expression with the minimum number of non-zero coefficients is the optimal FPRM-expression for f.*

Example 7 *The optimal polarity RMF-expression for function f in Example 6 corresponds to the polarity vector $\mathbf{H} = [2, 3]$, and is given by.*

$$\begin{aligned} f = {} & 2 \oplus {}^{3-*3}x_2 \oplus {}^{2-}x_1 \circ {}^{3-}x_2 \oplus {}^{2-*2}x_1 \oplus {}^{2-*2}x_1 \circ {}^{3-}x_2 \oplus 2\,{}^{2-*2}x_1 \circ {}^{3-*3}x_2 \\ & \oplus 2\,{}^{2-*3}x_1 \circ {}^{3-}x_2 \oplus 2\,{}^{2-*3}x_1 \circ {}^{3-*3}x_2. \end{aligned}$$

7 RMF-polarity Matrix

Similarly as in RM and GF expressions, an efficient way to determine the optimal polarity FPRMF-expression for a given function f is to calculate first the corresponding polarity matrix. Therefore, in this section we define polarity matrix for FPRMF-expressions for quaternary functions.

Definition 11 *The RMF polarity matrix* $\mathbf{P}_{RMF}$ *for* $f \in \mathbf{J}$ *is a* $(4^n \times 4^n)$ *matrix whose the i-th row consists of the coefficients in the FPRMF-expression for* f*, for the polarity vector* $\mathbf{H} = [i_1, \cdots, i_n]$ *where* $(i_1, \cdots, i_n)$ *is the quaternary representation of* i*.*

Example 8 *The RMF polarity matrix for function* f *in Example 6 is given by*

$$\mathbf{P}_{RMF} = \begin{bmatrix}
0 & 3 & 1 & 3 & 1 & 1 & 2 & 2 & 0 & 3 & 2 & 2 & 2 & 2 & 2 & 2 \\
1 & 2 & 2 & 1 & 0 & 3 & 0 & 2 & 1 & 1 & 0 & 2 & 0 & 0 & 0 & 2 \\
3 & 0 & 1 & 1 & 1 & 3 & 2 & 2 & 0 & 1 & 2 & 2 & 0 & 0 & 2 & 2 \\
3 & 3 & 0 & 3 & 2 & 1 & 0 & 2 & 3 & 3 & 0 & 2 & 0 & 2 & 0 & 2 \\
3 & 2 & 3 & 1 & 1 & 2 & 0 & 0 & 2 & 1 & 0 & 0 & 2 & 0 & 2 & 2 \\
1 & 3 & 2 & 3 & 3 & 2 & 0 & 0 & 1 & 1 & 0 & 0 & 2 & 2 & 0 & 2 \\
2 & 1 & 3 & 3 & 1 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 2 & 2 \\
1 & 2 & 0 & 1 & 3 & 2 & 0 & 0 & 3 & 1 & 0 & 0 & 2 & 0 & 0 & 2 \\
2 & 0 & 3 & 1 & 3 & 1 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\
2 & 1 & 2 & 3 & 2 & 1 & 0 & 0 & 3 & 3 & 0 & 2 & 3 & 3 & 3 & 2 \\
1 & 3 & 3 & 3 & 1 & 1 & 0 & 0 & 0 & 3 & 2 & 2 & 0 & 0 & 2 & 2 \\
2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 2 & 0 & 2 & 0 & 2 \\
3 & 3 & 3 & 1 & 3 & 0 & 2 & 2 & 2 & 3 & 0 & 0 & 2 & 0 & 2 & 2 \\
0 & 0 & 2 & 3 & 3 & 2 & 0 & 2 & 3 & 3 & 0 & 0 & 2 & 2 & 0 & 2 \\
0 & 2 & 3 & 3 & 1 & 2 & 2 & 2 & 0 & 3 & 0 & 0 & 0 & 2 & 2 & 2 \\
2 & 3 & 0 & 1 & 3 & 0 & 0 & 2 & 1 & 3 & 0 & 0 & 2 & 0 & 0 & 2
\end{bmatrix}.$$

8 Calculation of RMF Polarity Matrix

Harking and Moraga in [6] gave a method for the calculation of polarity matrices $\mathbf{P}_{GF}$ for GF-expressions of ternary functions. Their method starts from the truth-vector $\mathbf{F}$ of f. Unlike to that, Falkowski and Rahardja proposed method for calculation of polarity matrices $\mathbf{P}_{GF}$ for GF-expressions of quaternary functions starting from zero-polarity GF-expression coefficients vector [3]. In this section, we give two recursive methods for FPRMF polarity matrix calculation. The first method, named "recursion by columns", starts from the truth-vector $\mathbf{F}$ while the other named "recursion by rows", starts from the zero-polarity RMF-coefficient vector $\mathbf{A}$.

Recursion by columns

Now, we will construct the recurrence matrix relation for RMF polarity matrix calculation using proposed formal method. First, we define matrix $\mathbf{B}$.

Definition 12 *For an n-variable quaternary function* $f(x_1, x_2, \cdots, x_n)$ *the* $(4^n \times 4^n)$ *matrix* $\mathbf{B}$ *is defined as* $\mathbf{B} = \left[B^{i \oplus j}\right]$*, where* $\oplus$ *is the operation addition modulo 4.*

$$\mathbf{B} = \begin{bmatrix} B^0 & B^1 & B^2 & B^3 \\ B^1 & B^2 & B^3 & B^0 \\ B^2 & B^3 & B^0 & B^1 \\ B^3 & B^0 & B^1 & B^2 \end{bmatrix}.$$

Based on matrix $\mathbf{B}$ we generate the recursive square matrix $\mathbf{Q}_n$.

$$\mathbf{Q}_n = \left(\mathbf{R}_1^{-1} \cdot \mathbf{B}\right)^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 1 & 3 & 3 \end{bmatrix} \cdot \begin{bmatrix} B^0 & B^1 & B^2 & B^3 \\ B^1 & B^2 & B^3 & B^0 \\ B^2 & B^3 & B^0 & B^1 \\ B^3 & B^0 & B^1 & B^2 \end{bmatrix}$$

$$= \begin{bmatrix} B^0 & B^0 + 3B^1 & B^0 + 2B^1 + B^2 & B^0 + B^1 + 3B^2 + 3B^3 \\ B^1 & B^1 + 3B^2 & B^1 + 2B^2 + B^3 & B^1 + B^2 + 3B^3 + 3B^0 \\ B^2 & B^2 + 3B^3 & B^2 + 2B^3 + B^0 & B^2 + B^3 + 3B^0 + 3B^1 \\ B^3 & B^3 + 3B^0 & B^3 + 2B^0 + B^1 & B^3 + B^0 + 3B^1 + 3B^2 \end{bmatrix}$$

$$= \begin{bmatrix} Q_{n-1}^0 & Q_{n-1}^0 + 3Q_{n-1}^1 & W_{13} & W_{14} \\ Q_{n-1}^1 & Q_{n-1}^1 + 3Q_{n-1}^2 & W_{23} & W_{24} \\ Q_{n-1}^2 & Q_{n-1}^2 + 3Q_{n-1}^3 & W_{33} & W_{34} \\ Q_{n-1}^3 & Q_{n-1}^3 + 3Q_{n-1}^0 & W_{43} & W_{44} \end{bmatrix}, \tag{9}$$

where

$$W_{13} = Q_{n-1}^0 + 2Q_{n-1}^1 + Q_{n-1}^2, \quad W_{14} = Q_{n-1}^0 + Q_{n-1}^1 + 3Q_{n-1}^2 + 3Q_{n-1}^3,$$
$$W_{23} = Q_{n-1}^1 + 2Q_{n-1}^2 + Q_{n-1}^3, \quad W_{24} = Q_{n-1}^1 + Q_{n-1}^2 + 3Q_{n-1}^3 + 3Q_{n-1}^0,$$
$$W_{33} = Q_{n-1}^2 + 2Q_{n-1}^3 + Q_{n-1}^0, \quad W_{34} = Q_{n-1}^2 + Q_{n-1}^3 + 3Q_{n-1}^0 + 3Q_{n-1}^1,$$
$$W_{43} = Q_{n-1}^3 + 2Q_{n-1}^0 + Q_{n-1}^1, \quad W_{44} = Q_{n-1}^3 + Q_{n-1}^0 + 3Q_{n-1}^1 + 3Q_{n-1}^2.$$

In this equation, Q_{n-1}^i $(i = 0, 1, 2, 3)$ is a square matrix, which is one order lower than the matrix Q_n.

Now we rewrite equation (9) in the usually used form [1,2,3,6].

Assume that the truth-vector $\mathbf{F}$ of $f \in \mathbf{J}$ is split into 4 subvectors of 4^{n-1} successive elements

$$\mathbf{F} = \left\{\mathbf{F}_{[n-1,0]}, \mathbf{F}_{[n-1,1]}, \mathbf{F}_{[n-1,2]}, \mathbf{F}_{[n-1,3]}\right\}.$$

Then, based on (9) RMF polarity matrix $\mathbf{P}_{RMF}$ for quaternary functions can be calculated by recursive method named "recursion by columns", given in Theorem 4.

Theorem 4 *The polarity matrix* $\mathbf{P}_{RMF}$ *for* $f \in \mathbf{J}$ *can be calculated as*

$$\mathbf{P}_{RMF} = \mathbf{Q}_n(\mathbf{F}).$$

$\mathbf{Q}_k$, $k = 1, \ldots, n$ *is determined by the following recurrence matrix relations*

$$\mathbf{Q}_k(\mathbf{F}_{[k,j]}) = \begin{bmatrix} \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,0]} & \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,0]} + 3\mathbf{F}_{[k-1,1]}) & W_{13} & W_{14} \\ \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,1]} & \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,1]} + 3\mathbf{F}_{[k-1,2]}) & W_{23} & W_{24} \\ \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,2]} & \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,2]} + 3\mathbf{F}_{[k-1,3]}) & W_{33} & W_{34} \\ \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,3]} & \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,3]} + 3\mathbf{F}_{[k-1,0]}) & W_{43} & W_{44} \end{bmatrix}, \quad (10)$$

where

$$\begin{aligned} W_{13} &= \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,0]} + 2\mathbf{F}_{[k-1,1]} + \mathbf{F}_{[k-1,2]}), \\ W_{14} &= \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,0]} + \mathbf{F}_{[k-1,1]} + 3\mathbf{F}_{[k-1,2]} + 3\mathbf{F}_{[k-1,3]}, \\ W_{23} &= \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,1]} + 2\mathbf{F}_{[k-1,2]} + \mathbf{F}_{[k-1,3]}), \\ W_{24} &= \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,1]} + \mathbf{F}_{[k-1,2]} + 3\mathbf{F}_{[k-1,3]} + 3\mathbf{F}_{[k-1,0]}, \\ W_{33} &= \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,2]} + 2\mathbf{F}_{[k-1,3]} + \mathbf{F}_{[k-1,0]}), \\ W_{34} &= \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,2]} + \mathbf{F}_{[k-1,3]} + 3\mathbf{F}_{[k-1,0]} + 3\mathbf{F}_{[k-1,1]}, \\ W_{43} &= \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,3]} + 2\mathbf{F}_{[k-1,0]} + \mathbf{F}_{[k-1,1]}), \\ W_{44} &= \mathbf{Q}_{k-1}(\mathbf{F}_{[k-1,3]} + \mathbf{F}_{[k-1,0]} + 3\mathbf{F}_{[k-1,1]} + 3\mathbf{F}_{[k-1,2]}. \end{aligned}$$

Proof: *The proof follows from the Kronecker product structure of RMF transform matrix* $\mathbf{R}(n)$. *Thanks to this structure, the columns of* $\mathbf{P}_{RMF}$ *can be expressed as the convolution of* f *with the corresponding columns of* $\mathbf{R}(n)$ *(Theorem 2). Then, the proof follows from the convolution properties of RMF-expressions [8].*

We define three auxiliary vectors

$$\begin{aligned} \mathbf{T}_{[k-1,1]} &= 3\mathbf{F}_{[k-1,0]}, \\ \mathbf{T}_{[k-1,2]} &= 2\mathbf{F}_{[k-1,2]}, \\ \mathbf{T}_{[k-1,3]} &= 2\mathbf{F}_{[k-1,3]}. \end{aligned}$$

Then, (10) can be written as

$$\mathbf{Q}_k\left(\mathbf{F}_{[k,j]}\right) = \begin{bmatrix} \mathbf{Q}_{k-1}(q_{11}) & \mathbf{Q}_{k-1}(q_{12}) & \mathbf{Q}_{k-1}(q_{13}) & \mathbf{Q}_{k-1}(q_{14}) \\ \mathbf{Q}_{k-1}(q_{21}) & \mathbf{Q}_{k-1}(q_{22}) & \mathbf{Q}_{k-1}(q_{23}) & \mathbf{Q}_{k-1}(q_{24}) \\ \mathbf{Q}_{k-1}(q_{31}) & \mathbf{Q}_{k-1}(q_{32}) & \mathbf{Q}_{k-1}(q_{33}) & \mathbf{Q}_{k-1}(q_{34}) \\ \mathbf{Q}_{k-1}(q_{41}) & \mathbf{Q}_{k-1}(q_{42}) & \mathbf{Q}_{k-1}(q_{43}) & \mathbf{Q}_{k-1}(q_{44}) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{Q}_{k-1}\left(\mathbf{F}_{[k-1,0]}\right) & \mathbf{Q}_{k-1}\left(q_{22} + q_{13}\right) & W_{13} & W_{14} \\ \mathbf{Q}_{k-1}\left(\mathbf{F}_{[k-1,1]}\right) & \mathbf{Q}_{k-1}\left(q_{32} + q_{23}\right) & W_{23} & W_{24} \\ \mathbf{Q}_{k-1}\left(\mathbf{F}_{[k-1,2]}\right) & \mathbf{Q}_{k-1}\left(q_{42} + q_{33}\right) & W_{33} & W_{34} \\ \mathbf{Q}_{k-1}\left(\mathbf{F}_{[k-1,3]}\right) & \mathbf{Q}_{k-1}\left(q_{41} + \mathbf{T}_{[k-1,1]}\right) & W_{43} & W_{44} \end{bmatrix}, \quad (11)$$

where

$$\begin{aligned} W_{13} &= \mathbf{Q}_{k-1}\left(q_{23} + q_{14}\right), & W_{14} &= \mathbf{Q}_{k-1}\left(q_{23} + q_{33}\right), \\ W_{23} &= \mathbf{Q}_{k-1}\left(q_{21} + q_{41} + \mathbf{T}_{[k-1,2]}\right), & W_{24} &= \mathbf{Q}_{k-1}\left(q_{33} + q_{43}\right), \\ W_{33} &= \mathbf{Q}_{k-1}\left(q_{11} + q_{31} + \mathbf{T}_{[k-1,3]}\right), & W_{34} &= \mathbf{Q}_{k-1}\left(q_{13} + q_{43}\right), \\ W_{43} &= \mathbf{Q}_{k-1}\left(q_{33} + q_{14}\right), & W_{44} &= \mathbf{Q}_{k-1}\left(q_{13} + q_{23}\right). \end{aligned}$$

The number of operations needed to calculate $\mathbf{P}_{RMF}$, is reduced significantly if we first calculate these auxiliary vectors $\mathbf{T}$, and then the vectors that are arguments of the matrices $\mathbf{Q}_{k-1}$, as given in (11).

Recursion by rows

If we know the zero-polarity RMF-coefficient vector $\mathbf{A}$ of f, then the RMF polarity matrix $\mathbf{P}_{RMF}$, can be calculated through the following "recursion by rows" method, given in Theorem 5. This "recursion by rows" can be induced from (9) if we apply the 3-th, 4-th and 5-th step in proposed method for generation recurrence relations for polarity matrices calculation.

Let the vector $\mathbf{A}$ of zero-polarity RMF-coefficients is split into 4 subvectors of 4^{n-1} successive elements, i.e.,

$$\mathbf{A} = \left\{\mathbf{A}_{[n-1,0]}, \mathbf{A}_{[n-1,1]}, \mathbf{A}_{[n-1,2]}, \mathbf{A}_{[n-1,3]}\right\}.$$

Theorem 5 *The RMF polarity matrix* $\mathbf{P}_{RMF}$ *for* $f \in \mathbf{J}$ *can be calculated as*

$$\mathbf{P}_{RMF} = \mathbf{Q}_n(\mathbf{A}),$$

where the following recursive matrix relations are used for the calculation of $\mathbf{Q}_k$, $k = 1, \ldots, n$:

$$\mathbf{Q}_k\left(\mathbf{A}_{[k,j]}\right) = \begin{bmatrix} \mathbf{Q}_{k-1}(t_{11}) & \mathbf{Q}_{k-1}(t_{12}) & \mathbf{Q}_{k-1}(t_{13}) & \mathbf{Q}_{k-1}(t_{14}) \\ \mathbf{Q}_{k-1}(t_{21}) & \mathbf{Q}_{k-1}(t_{22}) & \mathbf{Q}_{k-1}(t_{23}) & \mathbf{Q}_{k-1}(t_{24}) \\ \mathbf{Q}_{k-1}(t_{31}) & \mathbf{Q}_{k-1}(t_{32}) & \mathbf{Q}_{k-1}(t_{33}) & \mathbf{Q}_{k-1}(t_{34}) \\ \mathbf{Q}_{k-1}(t_{41}) & \mathbf{Q}_{k-1}(t_{42}) & \mathbf{Q}_{k-1}(t_{43}) & \mathbf{Q}_{k-1}(t_{44}) \end{bmatrix} \tag{12}$$

where

$$\begin{aligned}
t_{11} &= \mathbf{A}_{[k-1,0]}, & t_{12} &= \mathbf{A}_{[k-1,1]}, \\
t_{13} &= \mathbf{A}_{[k-1,2]}, & t_{14} &= \mathbf{A}_{[k-1,3]}, \\
t_{21} &= \mathbf{A}_{[k-1,0]} + 3\mathbf{A}_{[k-1,1]}, & t_{22} &= \mathbf{A}_{[k-1,1]} + 3\mathbf{A}_{[k-1,2]}, \\
t_{23} &= \mathbf{A}_{[k-1,2]} + 3\mathbf{A}_{[k-1,3]}, & t_{24} &= 2\mathbf{A}_{[k-1,2]} + \mathbf{A}_{[k-1,3]}, \\
t_{31} &= \mathbf{A}_{[k-1,0]} + 2\mathbf{A}_{[k-1,1]} + \mathbf{A}_{[k-1,2]}, & t_{32} &= \mathbf{A}_{[k-1,1]} + 2\mathbf{A}_{[k-1,2]} + \mathbf{A}_{[k-1,3]}, \\
t_{33} &= 3\mathbf{A}_{[k-1,2]} + 2\mathbf{A}_{[k-1,3]}, & t_{34} &= 3\mathbf{A}_{[k-1,3]}, \\
t_{41} &= \mathbf{A}_{[k-1,0]} + \mathbf{A}_{[k-1,1]} + 3\mathbf{A}_{[k-1,2]} & & \\
&\quad +3\mathbf{A}_{[k-1,3]}, & t_{42} &= \mathbf{A}_{[k-1,1]} + 3\mathbf{A}_{[k-1,2]} + 3\mathbf{A}_{[k-1,3]}, \\
t_{43} &= 3\mathbf{A}_{[k-1,2]} + 3\mathbf{A}_{[k-1,3]}, & t_{44} &= 2\mathbf{A}_{[k-1,2]} + 3\mathbf{A}_{[k-1,3]}.
\end{aligned}$$

$$\mathbf{Q}_k(\mathbf{A}_{[0,j]}) = \mathbf{A}_{[0,j]},\ j = 0, 1, 2, 3.$$

Similarly to the "recursion by columns" method, the number of additions and multiplications can be reduced significantly if, instead (12), we use the following

formula

$$\mathbf{Q}_k\left(\mathbf{A}_{[k,j]}\right) = \begin{bmatrix} \mathbf{Q}_{k-1}(q_{11}) & \mathbf{Q}_{k-1}(q_{12}) & \mathbf{Q}_{k-1}(q_{13}) & \mathbf{Q}_{k-1}(q_{14}) \\ \mathbf{Q}_{k-1}(q_{21}) & \mathbf{Q}_{k-1}(q_{22}) & \mathbf{Q}_{k-1}(q_{23}) & \mathbf{Q}_{k-1}(q_{24}) \\ \mathbf{Q}_{k-1}(q_{31}) & \mathbf{Q}_{k-1}(q_{32}) & \mathbf{Q}_{k-1}(q_{33}) & \mathbf{Q}_{k-1}(q_{34}) \\ \mathbf{Q}_{k-1}(q_{41}) & \mathbf{Q}_{k-1}(q_{42}) & \mathbf{Q}_{k-1}(q_{43}) & \mathbf{Q}_{k-1}(q_{44}) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{Q}_{k-1}\left(\mathbf{A}_{[k-1,0]}\right) & \mathbf{Q}_{k-1}\left(\mathbf{A}_{[k-1,1]}\right) & W_{13} & W_{14} \\ \mathbf{Q}_{k-1}\left(q_{11} + \mathbf{S}_{[k-1,1]}\right) & \mathbf{Q}_{k-1}\left(q_{42} + q_{14}\right) & W_{23} & W_{24} \\ \mathbf{Q}_{k-1}\left(q_{21} + q_{13} + \mathbf{S}_{[k-1,1]}\right) & \mathbf{Q}_{k-1}\left(q_{24} + q_{12}\right) & W_{33} & W_{34} \\ \mathbf{Q}_{k-1}\left(q_{11} + q_{42}\right) & \mathbf{Q}_{k-1}\left(q_{24} + \mathbf{S}_{[k-1,3]}\right) & W_{43} & W_{44} \end{bmatrix}.$$

where

$$\begin{aligned} W_{13} &= \mathbf{Q}_{k-1}\left(\mathbf{A}_{[k-1,2]}\right), & W_{14} &= \mathbf{Q}_{k-1}\left(\mathbf{A}_{[k-1,3]}\right), \\ W_{23} &= \mathbf{Q}_{k-1}\left(q_{13} + \mathbf{S}_{[k-1,3]}\right), & W_{24} &= \mathbf{Q}_{k-1}\left(q_{14} + \mathbf{S}_{[k-1,2]}\right), \\ W_{33} &= \mathbf{Q}_{k-1}\left(q_{23} + q_{44}\right), & W_{34} &= \mathbf{Q}_{k-1}\left(\mathbf{S}_{[k-1,3]}\right), \\ W_{43} &= \mathbf{Q}_{k-1}\left(q_{13} + q_{44}\right), & W_{44} &= \mathbf{Q}_{k-1}\left(\mathbf{S}_{[k-1,2]} + \mathbf{S}_{[k-1,3]}\right), \end{aligned}$$

$$\begin{aligned} \mathbf{S}_{[k-1,1]} &= 3\mathbf{A}_{[k-1,1]}, \\ \mathbf{S}_{[k-1,2]} &= 2\mathbf{A}_{[k-1,2]}, \\ \mathbf{S}_{[k-1,3]} &= 3\mathbf{A}_{[k-1,3]}. \end{aligned}$$

Calculation of the auxiliary vectors $\mathbf{S}$ precedes calculation of arguments in $\mathbf{Q}_{k-1}$ like in the previous method.

9 Calculation Complexity

In this section, the efficiency of the presented methods for calculation of RMF polarity matrix is estimated through the number of operations required to calculate $\mathbf{P}_{RMF}$ for a quaternary function. For comparison, we give the number of operations in the corresponding methods for GF-expressions.

There are few methods to calculate the polarity matrix for GF-expressions of quaternary functions. A direct calculation by definition of $\mathbf{P}_{GF}$ for GF-expressions of n-variable quaternary functions requires $11^n - 4^n$ additions and $\frac{2}{3}(11^n - 5^n)$ multiplications [5]. In FFT-like algorithms proposed in [5], the number of additions and multiplications is $7n4^{n-1}$ and $n4^n$, respectively. The recursive algorithm proposed by Falkowski and Rahardja in [3] requires $A_n = \frac{4}{3}(13^n - 4^n)$ additions and $M_n = \frac{3}{8}(3^n - 1)4^n$ multiplications.

By the analogy to GF-expressions, we considered few ways to calculate the RMF polarity matrix. In a direct implementation of (6), the number of required additions and multiplications is $A_d^n = 4^{2n}(4^n - 1)$ and $M_d^n = 4^{3n}$, respectively. The number of additions and multiplications required for the polarity matrix calculation with FFT-like algorithm is $A_{FFT}^n = \frac{3n}{2}16^n$ and $M_{FFT}^n = \frac{7n}{4}16^n$, respectively.

The computational cost of methods proposed in Section 8 is stated by the following theorem.

Table 5: The number of additions and multiplications in calculation of $\mathbf{P}_{RMF}$.

	direct		FFT-like	
n	A_d^n	M_d^n	A_{FFT}^n	M_{FFT}^n
1	48	64	24	28
2	3840	4096	768	896
3	258048	262144	18432	21504
4	16711680	16777216	393216	458752
5	1072693248	1073741824	7864320	9175040
6	6.87027e10	6.8719477E10	150994944	176160768
	recursion by columns		recursion by rows	
n	A_c^n	M_c^n	A_r^n	M_r^n
1	14	3	12	3
2	280	57	240	57
3	4704	903	4032	903
4	76160	13737	65280	13737
5	1222144	206823	1047552	206823
6	19568640	3105417	16773120	3105417

Theorem 6 *The number of additions required to calculate RMF polarity matrix for an n-variable quaternary function, by using the recursive matrix relation (11) (recursion by columns) is* $A_c^n = \frac{14}{12}(16^n - 4^n)$. *If the relation (13) (recursion by rows) is used, the number of additions is* $A_r^n = (16^n - 4^n)$. *In both cases, the same number of* $M_c^n = \frac{3}{11}(15^n - 4^n)$ *multiplications is required.*

For illustration of this theorem, the Table 5 shows the number of additions and multiplications in calculation of the RMF-polarity matrix for different values of the number n of variables for different methods. Figures 1 and 2 show the number of additions and the number of multiplications needed for the calculation of the RMF polarity matrix with different methods.

It is obvious that methods proposed in Section 8 are more efficient than direct computation or FFT-like methods for the calculation of the RMF polarity matrices. It is important to note that the efficiency of our method increases with the number of variables.

10 Conclusion

We have proposed a method for construction of recursive procedures for the polarity matrices calculation in polynomial logical function representation. As particular cases the recursive methods proposed in [3] and [6] can be derived by our method. Based on our method we have constructed two algorithms, denoted as "recursion by rows" and "recursion by columns", for generation of polarity matrices for RMF-expressions of quaternary functions. To estimate their efficiency, we determined the number of operations required in each of them, and provided a comparison to other algorithms for generation of $\mathbf{P}_{RMF}$, as well as to the corresponding algorithms for polarity matrix for GF-expressions.

We showed that the proposed algorithms are more efficient than both direct calculation of $\mathbf{P}_{RMF}$ and related FFT-like algorithms. An important feature is that the efficiency of the proposed algorithms grows with the number of variables n in the represented functions. For example, the ratio between the number of additions in direct calculation of $\mathbf{P}_{RMF}$ and "recursion by rows" method is 4^n. The corresponding ratio for multiplications is greater than $\frac{11}{3}4^n$.

Our method can be used for construction of recursive relations for polarity matrices calculation for any Kronecker product based expression of MV functions.

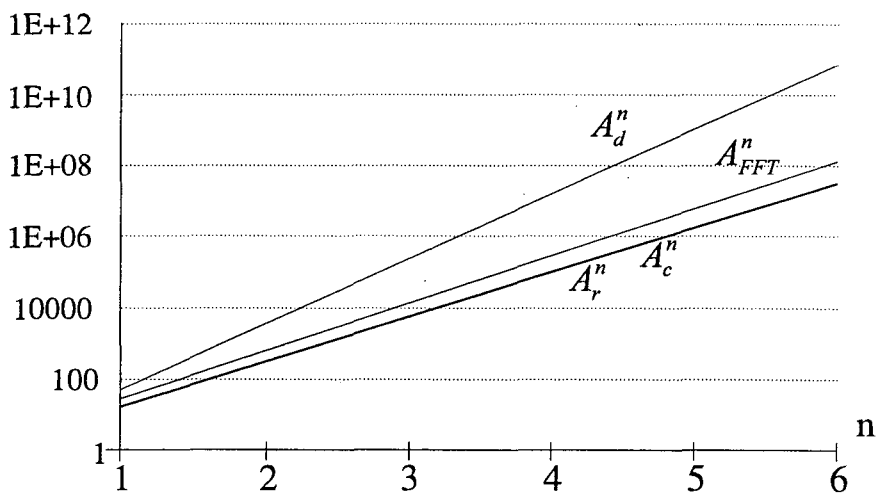


Figure 1: The number of additions needed for calculation $\mathbf{P}_{RMF}$.

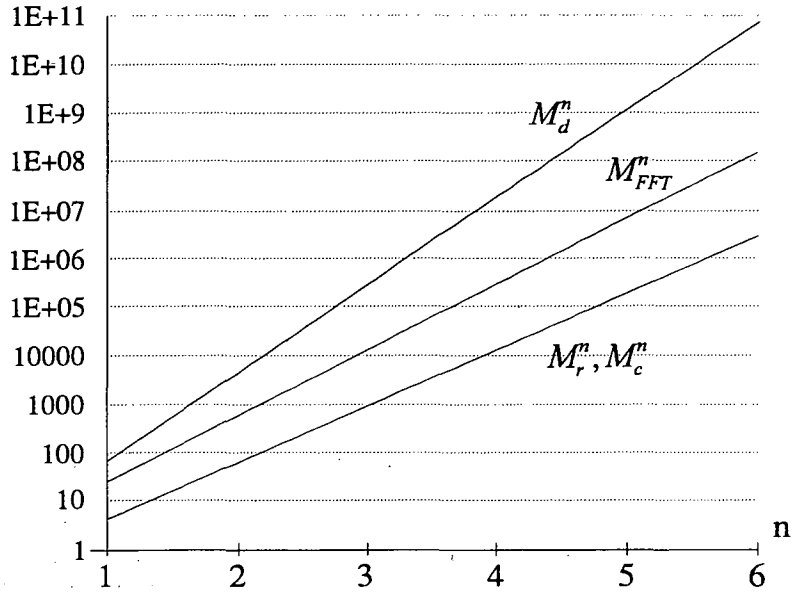


Figure 2: The number of multiplications needed for calculation $\mathbf{P}_{RMF}$.

References

[1] Falkowski, B.J., Rahardja, S., "Efficient algorithm for the generation of fixed polarity quaternary Reed-Muller expansions", *Proc. 25-th Int. Symp. on Multiple-Valued Logic*, Bloomington, 1995, 158-163.

[2] Falkowski, B.J., Rahardja, S., "Efficient computation of quaternary fixed polarity Reed-Muller expansions", *IEE Proc.-Comp.Digit.Tech.*, Vol.142, No. 5, 1995, 345-352.

[3] Falkowski, B.J., Rahardja, S., "Fast construction of polarity coefficient matrices for fixed polarity quaternary Reed-Muller expansions",*Proc. 5th International Workshop on Spectral Techniques*, Beijing, China, March 1994, 220-225.

[4] Falkowski, B.J., Rahardja, S., "Quasi-arithmetic expansions for quaternary functions",*Proc. IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, Chiba, Japan, 1995, 265-272.

[5] Green, D.H., "Reed-Muller expansions with fixed and mixed polarities over GF(4)", *IEE Proc.- Comp.Digit.Tech.*, Vol.137, No. 5, Sept. 1990, 380-388.

[6] Harking, B., Moraga, C., "Efficient derivation of Reed-Muller expansions in multiple-valued logic system", *Proc. 22nd IEEE Int. Symp. on Multiple-Valued Logic*, Sendai, Japan, 1992, 436-441.

[7] Muzio, J.C., Wesselkamper, T.C., *Multiple-valued Switching Theory*, Adam Hilger, Bristol, 1986.

[8] Stanković, R.S., Moraga, C., "Reed-Muller-Fourier representations of multiple-valued functions over Galois fields of prime cardinality",*Proc. IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, Hamburg, Germany, Sept. 1993, 115-124.

[9] Stanković, R.S., Janković, D., Moraga, C.,"Reed-Muller-Fourier versus Galois Field representations of Four- Valued Logic Functions",*Proc. 3rd Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, September 19-20, 1997. Oxford, UK, 269-278.

Acta Cybernetica **14** (1999) 285-302.

Object-Oriented Model for Partially Separable Functions in Parameter Estimation*

Jaakko Järvi†

Abstract

In parameter estimation, a model function depending on adjustable parameters is fitted to a set of observed data. The parameter estimation task is an optimisation problem, which needs a computational kernel for evaluating the model function values and derivatives. This article presents an object-oriented framework for representing model functions, which are partially separable, or *structural*. Such functions are commonly encountered, e.g., in spectroscopy.

The model is general, being able to cover a range of varying model functions. It offers flexibility at runtime allowing the construction of the model functions from predefined component functions. The mathematical expressions are encapsulated and a close mapping between mathematics and program code is preserved. Also, all interfacing code can be written independently of the particular mathematical formula. These properties together make it easy to adapt the model to different problem domains: only tightly controlled changes to the program code are required.

The paper shows how derivatives of the model function can be computed using automatic differentiation relieving the programmer from writing explicit analytical derivative codes.

The persistence of the objects involved is discussed and finally the computational efficiency of the function and derivative evaluation is addressed. It is shown that the benefits of the object-oriented model, namely the higher abstraction level and increased flexibility, are achieved with a very moderate loss of performance. This is demonstrated by comparing the performance with low-level tailored C-code.

1 Introduction

Even though object-orientation (OO) has become the dominating programming paradigm, it is quite slowly adopted to numerical applications, mainly because of the poor efficiency of OO programs in numerical codes. The progress in programming techniques and compilers is changing this situation and makes it possible to

*This work was supported by the Academy of Finland, grant 37178.

†Turku Centre for Computer Science, Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland, email: jaakko.jarvi@cs.utu.fi

take advantage of OO in numerical codes without a significant performance penalty [16]. This is demonstrated in this paper describing an OO model for parameter estimation of structural, partially separable functions.

The task of modelling data is commonly encountered in numerous application fields. The goal is to fit a model that depends on adjustable parameters to a set of observed data. A cost function, such as the sum of squared differences, is chosen to measure the agreement between the model and data. This function is minimised by adjusting the parameters of the model according to some optimisation algorithm.

The model can be based on some underlying theory about the data or be just a sum of convenient functions, such as polynomials. This article focuses on partially separable model functions, where the function is a sum of *component functions*, e.g., a spectrum consisting of a sum of spectral lines. The OO model presented in this article was developed while working on nuclear magnetic resonance (NMR) spectra estimation. Hence, the article includes a case study of NMR spectral fitting to make the ideas presented more concrete.

Numerous algorithms have been described for model fitting tasks in the literature [2, 14]. They are usually presented from the numerical analysis viewpoint, treating the model as a plain vector of parameters and a function for evaluating values and derivatives. However, this *flat* representation of the model function is not necessarily natural. The model may be structural consisting of several component functions, which possibly correspond to some real life entities. The function representation should be flexible. It should be possible to specify the composition of the component functions at runtime, rather than fix them in the program code. Furthermore, the function representation should be able to handle dependencies between parameters of different component functions. The flat model representation is therefore inconvenient for the user and it is the application developer's task to provide a conversion to and from the structural representation.

This article presents an OO model to serve as an intermediate link between the two representations described above. The model provides simultaneously an efficient computational kernel for the optimisation algorithms and the structured view for the user. It is a collection of classes comprising a core to represent structured model functions. These core classes implement the basic structural and flat views to the model function, as well as the mechanisms for function value and derivative calculations.

The extension of the core model for a specific application is done by providing a simple class for each type of component function. Essentially only member functions specifying the mathematical formulae of the component functions are required in these classes. Consequently, the particular mathematical expressions are encapsulated and the mathematical structures of the problem domain are preserved in the program code. This means that the necessary changes to program code are minor and well controlled if the model is applied to a different application area.

The model utilises the concept of *automatic differentiation* [15] for derivative computations. This relieves the programmer from writing analytical derivative codes. Automatic differentiation is made transparent to the programmer with operator overloading.

The core classes implement all the functionality needed for constructing component functions and their parameters. The user interface for this task can therefore be built solely based on the core classes. The addition of new classes to the model hierarchy does not cause any need for changes in the interfacing code. In section 3.5 we give an example of a user interface built in this manner.

This paper also discusses the computational efficiency and shows that the overhead arising from the higher abstraction level and greater runtime flexibility of the OO model is very moderate compared with a low-level C-code implementation. Persistence, i.e., the ability to store and retrieve the objects of the model is also considered.

The crucial parts of the model are presented using C++ language, but the model can be implemented in any language supporting inheritance, dynamic binding and operator overloading. However, the test runs were performed using a C++ implementation.

There are few descriptions of using object orientation together with parameter estimation in the literature. Related work can be found from [11, 17] containing general descriptions of computer systems sharing some similarities with our model. For description of an NMR analysis software built using a variant of the object oriented model presented here, see [10].

2 Parameter estimation problem

The task of fitting a parametric model function to a set of observed data points can be seen as minimisation of a cost function describing the distance between the model and the data. A common choice for the cost function is the sum of squares function. This least-squares model fitting problem can be stated as follows:

Let $y(x_i), i = 1, \ldots, m$ be a set of observed data points, $\mathbf{p} = (p_1, \ldots, p_k)$ be a vector of model parameters and $\hat{y}(x, \mathbf{p})$ a parameter-dependent model function. The maximum likelihood estimate of the parameters is obtained by minimising the chi-square function

$$\chi^2(\mathbf{p}) = \sum_{i=1}^{m} \left(\frac{y(x_i) - \hat{y}(x_i, \mathbf{p})}{\sigma_i} \right)^2, \tag{1}$$

where σ_i is the standard deviation of the measurement error of the ith data point. This formulation leads to a possibly non-linear optimisation problem which can be solved with iterative methods, most commonly with Levenberg-Marquardt or Gauss-Newton algorithms [2, 14]. The idea is to improve iteratively the trial solution

$$\mathbf{p}_{\mathbf{new}} = \mathbf{p}_{\mathbf{current}} + \boldsymbol{\Delta}\mathbf{p} \tag{2}$$

until an acceptable solution is found. The change $\boldsymbol{\Delta}\mathbf{p}$ is determined using the gradient and usually an approximation of the Hessian of the cost function. These in turn require calculation of the partial derivatives $\frac{\partial \hat{y}(x,\mathbf{p})}{\partial p_s}, s = 1, \ldots, k$ of the

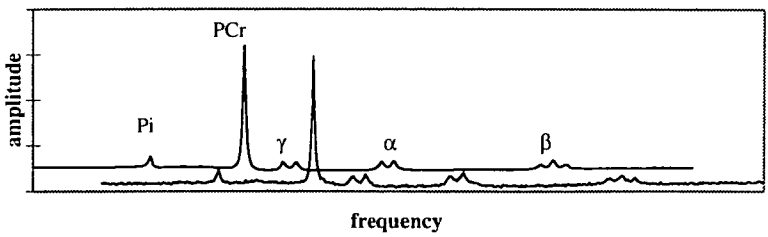


Figure 1: Example of a ^{31}P NMR spectrum (lower curve) and a model function (upper curve) fitted to the spectrum. α, β and γ peak groups originate from ATP molecules. The measured spectrum is shifted rightwards for clarity.

model function. Even though each iteration typically involves additional costs, such as solving a linear system of equations, the calculation of the model function and derivative values often dominate the overall cost.

The above clarifies the numerical view to the parametric estimation problem. The algorithms developed for the estimation must be supplied with the parametric model function, functions for the partial derivatives and the vector of modifiable parameters. Furthermore, $\hat{y}(x, \mathbf{p})$ is typically calculated at several points with constant $\mathbf{p}$. In cases we are interested in, $\hat{y}(x, \mathbf{p})$ is partially separable, that is, $\hat{y}$ can be represented as a sum of component functions $\hat{y}_j, j = 1, \ldots, n$, each being dependent on only r_j parameters, where $r_j << k$.

2.1 NMR spectroscopy case

In NMR spectroscopy, a signal of damping oscillations (FID) emitted by certain atomic nuclei (e.g. ^{31}P) is observed. An NMR spectrum is a Fourier transform of this signal. The spectrum contains peaks or resonance lines corresponding to nuclei in various compounds. The amplitude of a single peak is proportional to the number of equivalent nuclei resonating at that frequency. [6]

A typical ^{31}P NMR spectrum is shown in Fig. 1. Signals of inorganic phosphate (Pi), phosphocreatine (PCr) and adenosine triphosphate (ATP) can be identified from the spectrum. The aim is to find the amplitudes and frequencies of the identified compounds. This is done by quantifying the spectrum or the FID, which is represented as a superposition of parametric functions, each corresponding to a single resonance line. This parametric model function is fitted to the measured signal and the results, peak intensities and frequencies, are calculated from the model parameters. Fig. 1 also shows a fitted model.

Basically we have a structural model function consisting of a sum of component functions, the resonance lines. Several lineshapes are encountered, the most common being the Lorenz function described by amplitude A, frequency f, phase ϕ and damping factor d. A model of n reconance lines in a somewhat simplified form in time domain is then

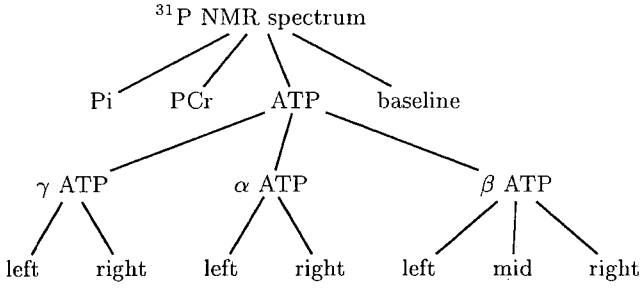


Figure 2: Example of a model function instance.

$$\hat{y}(t, \mathbf{p}) = \sum_{j=1}^{n} A_j \cos(2\pi f_j t + \phi_j) e^{-d_j t}, \tag{3}$$

where $\mathbf{p} = (A_1, f_1, d_1, \phi_1, \ldots, A_n, f_n, d_n, \phi_n)$. As can be seen, the sum function is partially separable. Note that, contrary to this simplified expression, the NMR signal can contain different lineshapes and there may be additional terms in the sum. [5]

Dependencies between parameters of different component functions are typical for NMR models. Consider the ATP molecule. It is known a priori that 7 peaks altogether originate from the ATP molecules. The peaks come in three groups: α, β and γ. These groups have equal amplitudes. The groups α and γ consist of two peaks each having again equal amplitudes. The β-group consists of three peaks with relative amplitudes $1 : 2 : 1$. The frequency differences between the peaks inside the groups are known and it is reasonable to assume that the damping factors of all the peaks are equal. Taking these into consideration, the amplitudes, damping factors and frequencies of 7 peaks are actually defined by only one amplitude, one damping factor and three frequency parameters. The hierarchical structure of ATP and other peaks in the NMR example spectrum is depicted in Fig. 2.

To sum up the problem setting, the estimation of the parameters of the function $\hat{y}$ is the task to be performed. This is done by minimising the chi-square error with respect to the measured signal, where the partial derivatives of $\hat{y}$ must be calculated repeatedly. Function $\hat{y}$ has a hierarchical structure corresponding to the peaks in the spectrum.

3 Object-oriented model

Significant savings in development time can be achieved with careful design of the model function representation. In the case of structural model functions, the utmost goal is flexibility. The number and type of the component functions may vary and there may be common or related parameters between the component functions.

The model function representation ought to be able to handle these situations with ease and yet be able to compute the function value and derivatives efficiently.

An important issue is the user interface for managing the model functions. The user constructs the model functions and observes or edits the model parameters. The programmer's task to provide this interface for varying models is considerably alleviated if the interface can be implemented without the need to know the actual types or number of the component functions. The term *user* refers to a human operator of a computer program whereas by *client* we denote the programmer or code calling the functions or using other services of the object-oriented model.

The object-oriented approach provides a convenient means to build a function representation to meet the requirements detailed above. The model consists of two separate class hierarchies, the *function hierarchy* and the *parameter hierarchy*. An essential component is also a library for automatic differentiation. The hierarchies are first discussed accentuating the client view to the classes and then the process of function value and derivative evaluation is clarified. While reading, the reader may consult the object diagram in Fig. 6 representing the NMR example as objects from function and parameter hierarchy.

3.1 Function hierarchy

The classes of the function hierarchy (Fig. 3) represent the component functions of the structural model function (the nodes of the tree in Fig. 2). The base of the hierachy is the abstract base class base_model, which defines the interface for the function classes; each function can compute the value and derivatives at a given point. The base_class maintains a vector of parameters and defines member functions for accessing them. Different component functions are derived from the base_model class. These can be either *elementary* or *composite* functions.

Composite functions maintain a list of other component functions. They simply group other components. A composite function computes its values and derivatives by calling the evaluation functions of its *child* functions. Each composite model *owns* the models in its child list. The top_model class represents the whole model function to be fitted and implements the interface to the client code. It also maintains the vector of the adjustable parameters used by the optimisation algorithm.

The generic elementary_model class encapsulates the common features of the component functions to make the derived classes as simple as possible. The template parameter of the generic class specifies the number of parameters in the function. We will return to the details of this template in section 4. Now it suffices to say that the elementary model holds the parameters of the mathematical function to be calculated as *automatically differentiable numbers* in the proxy data member.

Fig. 4 shows a complete class definition of an example class derived from elementary_model. These derived classes contain the actual mathematical formulae of the model function (the eval function). In addition, only two simple utility functions (create and get_class_name) are needed. These are the only requirements for each elementary function class and it is thus very easy to extend the function hierarchy to cover new function types.

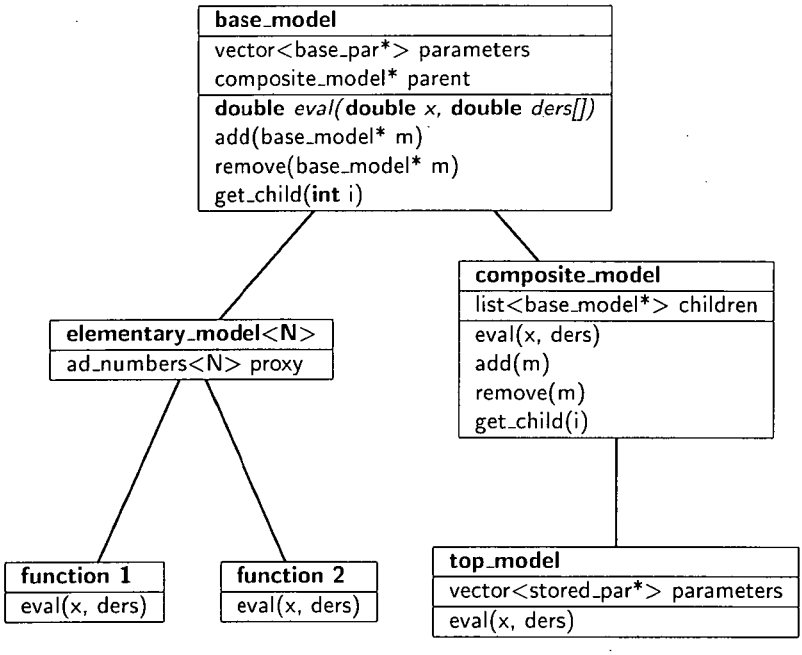


Figure 3: Model function class hierarchy.

Gamma et al. [8] have proposed some general methods for representing hierarchical structures in an object-oriented language. This model function hierarchy can be seen as a version of the *Composite design pattern.* Regarding the implementation issues of this pattern discussed by Gamma et al. we have chosen to maintain explicit parent references implemented as a pointer in the base_model class. We also chose to maximise the interface of the base_model. This means that, e.g., operations for manipulating the list of children of the composite models (add, remove) are declared and defined in base_model. This gives transparency for the client but on the other hand the operations do not have a meaning for elementary models. Therefore, by default, the operations add and remove fail (e.g. by raising an exception) and the functions are overridden in the composite_model class to give them meaningful definitions.

Not all functions are shown in the class diagram of Fig. 3. The base_model class also defines functions for adding and removing parameters as well as functions for naming the models. The *virtual constructor* [8, 1] mechanism is utilised in the object construction, requiring the two virtual functions, create and get_class_name, to be overridden in each derived class.

```
class lorenz : public elementary_model<4> {
public:
  lorenz* create() {return new lorenz(); }
  string get_class_name() {return "lorenz"; }
  enum {amp, freq, damp, ph };
  double eval(double x, vector<double>& ders) {
    return store_derivatives( ders,
      par(amp)*cos(2*pi*par(freq)*x + par(ph)) * exp(-x*par(damp))); }
};
```

Figure 4: Definition of an example function derived from the elementary_model class.

3.2 The parameter hierarchy

The parameter of a model function is basically just a value of some floating point type. However, the same parameter value may be shared by several component functions or there may be other dependencies between parameters. Hence, not all parameters of the component functions store a value. As a consequence, just representing a parameter as a floating point number is not sufficient to allow the component models to use the parameters in a uniform way. Therefore parameters are represented as classes from the parameter hierarchy (Fig. 5).

The base_par class is the topmost class of the hierachy and provides the common interface, the functions get_value and get_derivative for retrieving the value and initial derivative of the parameter. The stored_par class represents actually stored, adjustable parameters. The dependent_par class is the base class for dependent parameters and linear_par is for expressing linear relations between parameters. Other dependencies may be implemented by deriving new classes from the dependent_par class.

Each dependent parameter holds a pointer to another parameter, a *parent* parameter. The value is resolved by asking the value of the parent recursively until finally an instance of a stored_par class will end the recursion. The same mechanism applies for derivatives. The get_derivative function evaluates the derivative with respect to the underlying stored parameter. For stored_par this is 1 (the derivative of a variable with respect to itself is 1), while for linear parameters we get it by multiplying the derivative of the parent with the linear factor (see the code outlined in Fig. 5).

All parameters also maintain a child list and a pointer to the model function owning the parameter. The dependent parameters contain a vector of *parameter modifiers* such as the coefficients of the linear relation. The number of these modifiers is fixed for each derived class and given in the constructor.

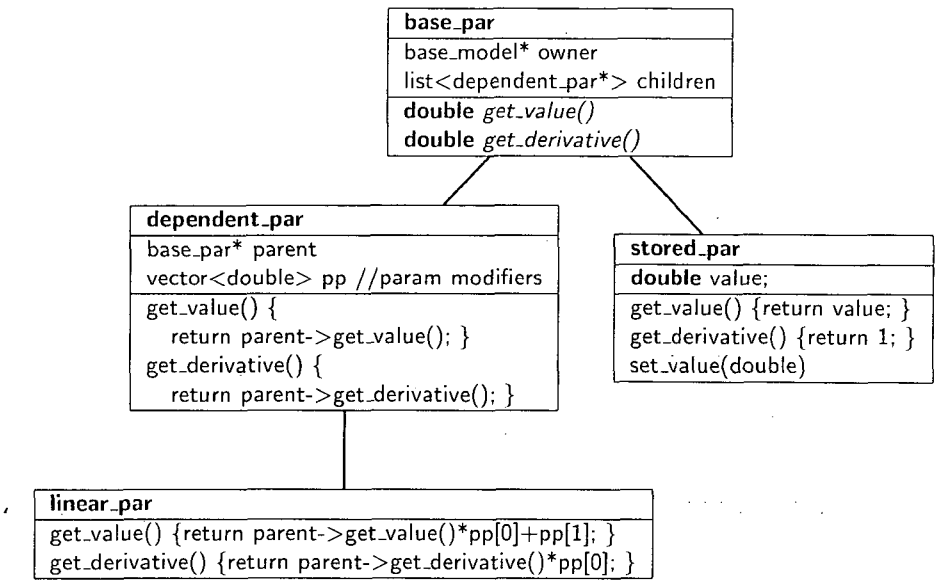


Figure 5: Parameter hierarchy.

3.3 Enforcing the consistency

The data structure for representing structural functions consists of several objects from the function and parameter hierarchies (Fig. 6). It is a combination of two object trees, both maintaining child node lists and parent pointers. In addition, the nodes of the model tree may own nodes of the parameter tree. This relation is represented as a list in the model tree node and a corresponding owner pointer in the parameter tree node. Furthermore, a vector of references to the adjustable parameters in the parameter hierarchy is maintained in the topmost model function.

To be able to guarantee the consistency of such a complex structure the construction and manipulation of the objects involved in the data structure must be controlled tightly. Though not shown in the class definitions, the creation and destruction of models is not part of the public interface of the classes, instead the creation of the objects is delegated to a special *creator* object and the destruction is performed from within the member functions of the classes of the hierarchy.

The final data structure maintains several invariances. The child list of a composite model is kept consistent with the parent references of the children. The same applies to child/parent relation in the parameter hierarchy as well as the parameter/owner relation between model functions and parameters.

The relation between parameters and models is further restricted. A parameter and its descendant can not be owned by different function hierarchies. Furthermore, the owner function of a dependent parameter must be a descendant of the owner of the parent parameter. Some of the invariances are guaranteed automatically by

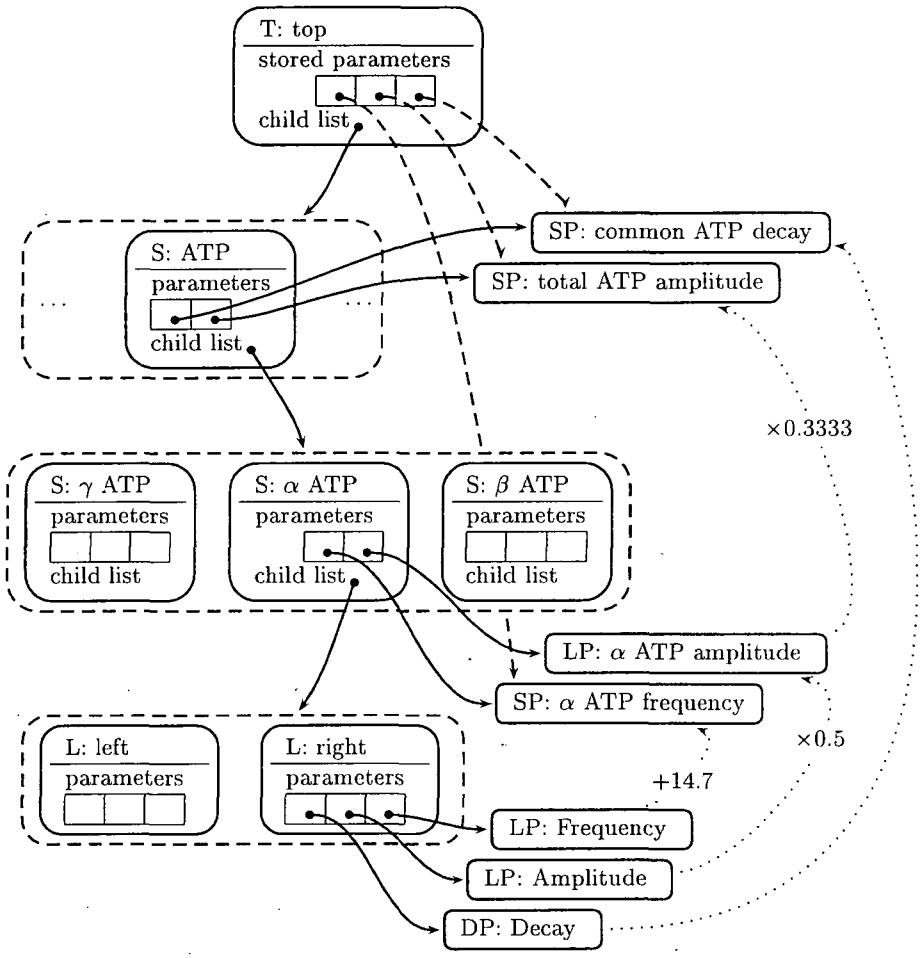


Figure 6: Instantiated objects and their relations illustrated in the NMR case (part of the ATP molecule). Solid lines represent ownership relation, while dashed lines are non-owning pointers. Dotted lines are parent links. The class of each object is given in parenthesis (C=composite, L=lorenz, LP=linear parameter, DP=dependent parameter, SP=stored parameter). Along the parent links of the parameters are the formulas for computing the values of linear parameters.

the restricted object construction. Others are enforced by raising an exception if a user tries to perform an operation which conflicts with an invariance condition.

3.4 Constructing model functions

The starting point of a model function is an instance of the top_model class. After it has been created, component models can be added to its child list.

The construction of objects is delegated to a special creator object implementing the virtual construction mechanism. The purpose of this is to make the client code, which initiates the creation of objects, independent of the changes in the elementary function classes.

The class of the object to be created is specified as a class name string at runtime. This is a convenient way of initiating object construction. Since the object creation task is most likely initiated by a user command, it is quite natural to specify the class as a class name string. The user may, e.g., have selected the class from a selection list.

The creation mechanism requires each class to *register* itself (one line of code) and define the virtual functions get_class_name and create. Otherwise the creation mechanism is totally independent of the derived classes: e.g., adding new elementary functions to the model hierarchy does not have any effect on the client code. For details of the virtual construction mechanism, see [8] describing several creational design patterns.

3.5 Model editor

As an example of a user interface for specifying structural model functions, Fig. 7 shows a snapshot of the model editor we have written. The structural function tree is visible on the left and the parameters of the currently selected model on the right. The names of the functions as well as the parameter names and values can be edited freely on the spot. There are buttons and menu commands for adding and removing functions and parameters, defining relations between parameters and storing and retrieving models. As pointed out above, the code of the model editor is totally independent of the particular elementary function classes derived from the model function hierarchy.

3.6 Persistence of model objects

In addition to methods for creating and modifying the structural model functions, means for their storage and retrieval are needed. In object-oriented systems, the ability of objects to live beyond the lifetime of the program is called persistence. It can be achieved using *object serialisation*, the common approach used in commercial class libraries such as MFC [12] and OWL [13]. This approach relies on virtual construction mechanism and requires the programmer to specify reading and writing methods for each persistent class.

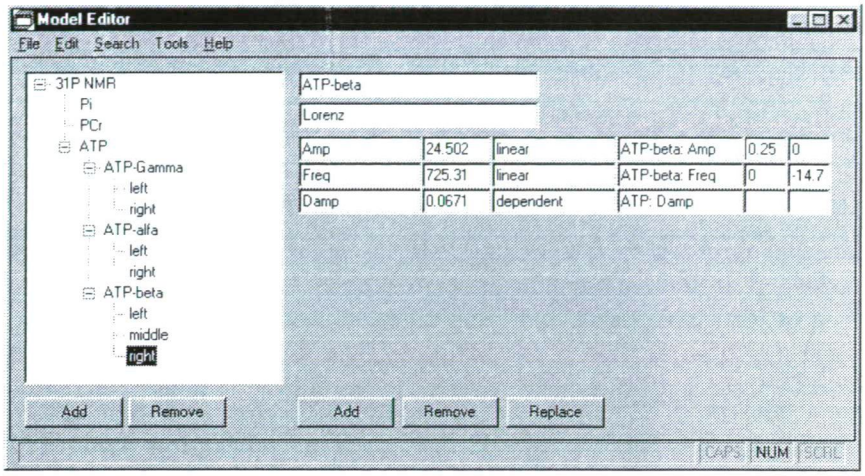


Figure 7: Snapshot of our model editor.

Virtual construction is already included in the model and parameter hierarchies. Furthermore, when the model hierarchy is extended, no new data members need to be introduced in the derived elementary function classes. Therefore the read and write functions can be inherited and need not be specified. Consequently, the implementation of persistence can be encapsulated entirely into the core classes of the model and no changes are required when new classes are added to the hierarchies.

4 Evaluation of function values and derivatives

In this section, the function value and derivative computation in our model is explained. The concept of automatic differentiation is described and it is shown how to calculate the derivatives of structural functions easily and yet effectively with this technique.

4.1 Automatic differentiation

The derivatives are traditionally calculated either symbolically or by using divided differences. The former may be quite difficult and error-prone while the latter introduces truncation errors and may be inaccurate and inefficient. Automatic differentiation provides an appealing alternative.

In automatic differentiation, the derivatives are computed by the well-known chain rule, but instead of propagating symbolic functions, numerical values are propagated along the computation. The evaluation of the function and its derivatives are calculated simultaneously using the same expressions. There are several descriptions about automatic differentiation [15, 1, 3] and also software packages

available [9, 4]. Some packages preprocess the source code to add the necessary statements for computing the derivatives. Other packages, using programming languages that support operator overloading, implement the differentiation as a class library without the need for a separate precompilation.

There are interesting computational issues concerning the implementation of automatic differentiation. The chain rule can be used either in *forward* or *backward* mode or in something between. The implementation involves a tradeoff between time and space complexity. In this article the forward mode automatic differentiation is used. It is simple and fits very well in this particular application as will become clear below.

In forward mode automatic differentiation, instead of computing with scalar values, we compute with automatically differentiable numbers (ADN) $\langle \mathbf{f}, \nabla \mathbf{f} \rangle$. An ADN consists of a value and a vector of partial derivatives of a function at a given point. When building expressions with these objects, at the leaf level of the expression tree $\mathbf{f}$ is either a variable or a constant. When differentiating with respect to N variables, the derivative of the ith variable is represented as the ith canonical unit vector of length N and the derivative of a constant with a zero vector. For example, when differentiating with respect to three variables x, y, z the constant 3.14 is expressed as $\langle 3.14, (0,0,0) \rangle$ and the variable y as $\langle y, (0,1,0) \rangle$. Computation with these objects utilises the chain rule of derivatives.

$$\frac{\partial}{\partial t} f(g(t))\bigg|_{t=t_0} = \left(\frac{\partial}{\partial s} f(s)\bigg|_{s=g(t_0)} \right) \left(\frac{\partial}{\partial t} g(t)\bigg|_{t=t_0} \right) \tag{4}$$

As an example, consider the two-derivative case for function $y+\sin(x^2)$. Starting with $\langle y, (0,1) \rangle + \sin\langle x, (1,0) \rangle^2)$ by squaring x, we get $\langle y, (0,1) \rangle + \sin(\langle x^2, (2x, 0) \rangle)$. Taking the sine gives $\langle y, (0,1) \rangle + \langle \sin(x^2), (2x\cos(x^2), 0) \rangle$ and finally the addition with y gives $\langle y + \sin(x^2), (2x\cos x^2), 1) \rangle$. For numerical work, the computation is not done symbolically, rather the actual values of the function and its derivatives are calculated and propagated through the expression. Given $x = 2, y = 4$ the same example becomes

$$\begin{aligned} \langle 4, (0,1) \rangle + \sin(\langle 2, (1,0) \rangle^2) &= \langle 4, (0,1) \rangle + \sin(\langle 4, (4,0) \rangle) = \\ \langle 4, (0,1) \rangle + \langle 0.06976, (1.9951, 0) \rangle &= \langle 4.06976, (1.9951, 1) \rangle. \end{aligned}$$

The method can be applied to any machine-computable function. All that is needed is to code the differentiation rules for simple functions and operations. Then any function composed of those elementary functions can be differentiated automatically. In C++ this means overloading common functions and operators for objects described above.

The forward mode automatic differentiation for calculating gradients can be computationally unattractive if applied blindly. If the gradient has n elements, the computation may require up to order of n as much time as computating the value of the same expression. However, in the case of partially separable functions the

forward mode can be applied efficiently. If we consider the model function as a whole, it may have quite a number of parameters, but the number of parameters of the individual elementary functions is typically rather low and known beforehand. Furthermore, different elementary functions are only related via a summation expression, which means that also the derivatives are just summed together. Consequently, we use automatic differentiation in computing the local gradients of the elementary functions and update the calculated values via pointers to the common derivative vector.

The computing time of the local gradients can be further reduced. By using C++ templates, moderate size derivative vectors of ADNs can be replaced with special sparse vectors to yield very efficient code [10]. This method was utilised in the test runs described in section 4.4.

4.2 The function evaluation process

The model function is evaluated by calling the eval function of the topmost class, which will traverse all the contained models and calculate their cumulative values at a given point. The derivatives are computed simultaneously using automatic differentiation. The derivative vector is passed as a parameter to the eval function. First the resulting derivative vector or gradient is initialised to zero. Each elementary function reads the values and initial derivatives of the parameters (with the get_value and get_derivative functions) and constructs ADNs from them. If the elementary function has n parameters, ADNs having n-dimensional derivative vectors are used. The mathematical expression is then evaluated using ADNs and each elementary function updates the resulting derivative values to the actual gradient vector. This is accomplished with a call to store_derivatives function defined in the elementary_model template (see Fig. 4), which adds the derivative values to the right positions of the gradient.

After the whole function tree has been traversed, the function value is returned and the gradient is available as the derivative vector passed to the eval function.

4.3 Computational efficiency

With regard to the computational efficiency the evaluation strategy includes a few pitfalls. Firstly, dynamic binding is applied in the eval function invocations. There is an inherent additional cost in a call to a dynamically bound virtual function compared with a statically bound function [7]. Furthermore dynamic binding precludes the use of inlined functions. Inline expansion can speed up function calls and is beneficial for small functions. However, in this case the computational cost of the function call is probably minor compared to the cost arising from the evaluation of the actual mathematical formulae of the elementary functions, recalling that the derivatives are also calculated in the same function. Considering this, the relative cost of the slightly slower function call is most likely insignificant in this case.

Secondly, dynamic binding is also applied between parameters in the get_value and get_derivative functions. In this case the extra cost may be notable. The

evaluation of an elementary function having n parameters would yield at least n virtual function calls to fetch the parameter values. The number of calls is larger if dependent parameters are involved. However, in model fitting tasks, the model function is evaluated repeatedly at several points, without changing the parameter values. Taken the example from NMR spectroscopy, the region of interest may contain thousands of points. Therefore the parameter values can be cached and only when the parameter values are changed, each elementary function reads the values and derivatives with the virtual get_value and get_derivative functions and stores the values to local proxy variables (ADNs). With this approach the relative cost of retrieving the parameter values via virtual functions is of little consequence. The caching is made transparent to the client code by maintaining a flag in the topmost class indicating whether the values in the proxy variables are valid or not.

Also, the updating of the local gradients to the global derivative vector must be efficient. This is implemented in the elementary_model template by maintaining a mapping from each local parameter index to an index in the global derivative vector. These mappings can be constructed prior to the first model evaluation. In this task the function tree must be traversed once, but this causes no efficiency problems, since the indices only change if the model function changes, i.e., new component functions are added or removed. At evaluation time the only additional cost is an extra indirection for each parameter.

Some cost may also arise if the composite models in the function tree contain many levels (e.g. in the ATP compound). From the computational point of view, it is not necessary to traverse all composite functions during the evaluation, rather it is sufficient to call the evaluation functions of the elementary models directly and save the cost of a few virtual calls. This is easy to implement by maintaining a separate list of the leaf nodes in the top_model class, which we did in the test runs.

4.4 Test runs

To assess the efficiency of the model some test runs were performed. As a test case, we used formulae from the NMR case consisting of 10 component functions having 24 adjustable parameters altogether. Five different alternatives to perform the function and derivative computations were programmed:

1. A tailored low-level C-code with analytical derivatives.
2. The presented OO model with analytical derivatives.
3. The OO model with automatic differentiation.
4. A straightforward OO implementation, without any caching.
5. A low-level implementation of the function with finite difference value approximations of the derivatives.

In the tailored low-level implementation, the model function was totally fixed at design time, so any change in the function requires changes in the code. The code

Implementation	Relative time
1. Tailored low level C-code	1.00
2. OO model with analytical derivatives	1.07
3. OO model with automatic differentiation	1.29
4. Straightforward OO implementation	2.48
5. Divided difference approximations	16.52

Table 1: Relative evaluation times of the different methods for computing the value and derivatives of the NMR model.

was hand-optimised to a reasonable level (not making any processor specific tricks). All subexpressions were calculated only once and all relations between parameters were directly written into the code as effectively as possible. It is fair to say that the code used was as fast as possible.

In the second case, the OO model presented was used, but the derivatives of the elementary functions were calculated analytically. This case should roughly represent the extra cost originating from the dynamic binding of the model functions, as well as the cost arising from not coding the dependencies between the parameters directly.

In the third case, the OO model was used with derivatives computed using automatic differentiation. Table 1 shows the results and confirms the extra cost being quite acceptable compared with the flexibility the model offers.

In the fourth case, no proxies for parameters were used, rather the initial values and derivatives were retrieved during each evaluation using the virtual function invocations. This demonstrates that the performance may drop significantly if the programmer is not aware of the principles affecting efficiency in OO programs.

In the fifth case, derivatives were approximated with divided difference values. The benefit of this alternative is that only the code for evaluating the value of the model function is needed. The performance is, however, very poor requiring n evaluations of the model function, where n is the number of elements in the gradient. Also, the accuracy is harder to assess.

The test runs were performed under Linux on Intel Pentium processor. The C++ compiler used was KAI C++ 3.2.d with optimisation flags +K2 -O3.

5 Conclusions

An object-oriented model for parameter estimation of partially separable function was described. The model achieves two goals. Firstly it gives an easily extendible OO framework for representing partially separable functions in a structured way, resembling the physical real-life interpretation and mathematical structure of the functions. Secondly, it offers an interface to an optimisation algorithm, namely a vector of adjustable parameters and a function capable of computing the value and derivatives of the model function efficiently.

To achieve the first goal, the model separates the commonalities of partially separable functions from the specific mathematical formulae. The formulae are encapsulated to a few very simple classes. It is therefore easy to apply the model to different problem domains, since changing these classes or adding new ones to the model does not affect the client code using the model. Furthermore, relations between parameters are handled by the model and they do not complicate the mathematical expressions of the component functions.

The derivatives needed in the parameter estimation are obtained using automatic differentiation. Hence, there is no need to hand-code analytical derivatives or use divided difference values.

Considering the second goal, the calculation of function values and derivatives is efficient. In our example case from NMR spectroscopy, the evaluation of the OO model required only 29% more time than a low-level tailored implementation of the same function. As a compensation, in the OO model the final function as well as relations between parameters can be specified at run-time, the model is easily extendible to cover new component functions and no hand-coded derivatives are required.

To sum up, the paper gives practical guidelines for implementing an efficient OO computational kernel for partially separable functions. With an example, we showed that OO programming offers substantial benefits, such as higher abstraction level, code reuse, flexibility and handling of complexity for numerical programming as well. Furthermore, the advantages can be achieved with a moderate loss of performance.

References

[1] Barton J. J., Nackman L. R.: Scientific and Engineering C++, Addison-Wesley, Reading Massachusetts 1994.

[2] Bazaraa M.S., Sherali H.D., Shetty C. M.: Nonlinear Programming: Theory and Algorithms, 2nd Edition, Wiley 1993.

[3] Editors: Berz M., Bischof C. H., Corliss G. F., and Griewank A.: Computational Differentiation - Techniques, Applications, and Tools,.SIAM, Philadelphia Pennsylvania 1996.

[4] Bischof C. H., Carle A., Corliss G. F., Griewank A., Hovland P.: ADIFOR: Generating derivative codes from Fortran programs, Scientific Programming, 1 (1992) 1-29.

[5] Bovée W. M. M. J.: Quantification in in vivo NMR, Spectral editing, in: Magnetic Resonance Spectroscopy in Biology and Medicine, eds. de Certaines J. D., Bovée W. M. M. J., Podo F., 181-207, Pergamon, Oxford 1992.

[6] Derome A. E.: Modern NMR Techniques for Chemistry Research, 63-90, Pergamon, Oxford 1991.

[7] Driesen K., Hölzle U.: The Direct Cost of Virtual Function Calls in C++, ACM Sigplan Notices, OOPSLA'96 Proceedings, **31** (1996) 306-323.

[8] Gamma E., Helm R., Johnson R., Vlissides J: Design Patterns, Elements of Reusable Object-Oriented Software, Addison-Wesley, Reading Massachusetts 1995.

[9] Griewank A., Juedes D., Utke J.: ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++, ACM Transactions on Mathematical Software, **22** (1996) no.2, 31-167.

[10] Järvi J.: A PC program for automatic analysis of NMR spectrum series, Computer Methods and Programs in Biomedicine **52** (1997) 213-222.

[11] Majoras R. E., Richardson W. M., Seymour R. S.: An object-oriented approach to evaluating multiple spectral models, Journal of Radioanalytical and Nuclear Chemistry **193** (1995) 207-210.

[12] Microsoft, Microsoft Foundation Class Library, Microsoft Corporation.

[13] Borland, Borland C++ 5 Programmer's guide, Borland International, 1996.

[14] Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P.: Numerical Recipes in C: the Art of Scientific Computing, 2nd Edition, Cambridge University Press, New York 1992.

[15] Rall L.B.: Automatic differentiation: Techniques and Applications, Lecture Notes in Computer Science 120, Springer-Verlag, Berlin 1981.

[16] Robinson A.D.: C++ Gets Faster for Scientific Computing, Computers in Physics **10** (1996) 458-462.

[17] van Tongeren B. P. O., Boxman R. D. C., Deumens J. W., van Leeuwen J. P., Mehlkopf A. F., van Ormondt D., de Beer R.: QUANSIS, An object-oriented data-analysis system for in vivo NMR signals, Journal of Magnetic Resonance Analysis, **2** (1996) 75-84.

Acta Cybernetica **14** (1999) 303-314.

Hausdorff Dimension of Univoque Sets

Gábor Kallós *

Dedicated to Professor Imre Kátai on his 60th birthday

Abstract

In this paper we present the results obtained so far for the determination of the Hausdorff dimension of the univoque set, in number systems with base number greater than 1. The investigation is based on the methods presented in [1] and [2]. We illustrate the theoretical results with interesting examples.
Keywords: Number Theory, Expansions of Numbers, Univoque Sequences

1 Introduction

A lot of interesting problems arose relating to the number systems. A collection of these is presented in D. E. KNUTH's significant book [5].

In this paper we investigate the problem, what the numbers are, the representation of which is unique in a number system. The answer to this question derives immediately, when the base number of the number system is a positive integer. For example, in the decimal system a number has a unique representation if its form is an infinite decimal fraction, except the numbers with a pure nine tail $(999\ldots)$. The integers, the finite decimal fractions and the infinite decimal fractions with the tail $999\ldots$ have two representations.

$\pi = 3,1415926\ldots$ - only one representation

$3,14 = 3,13999\ldots$ - two representations

Essentially the same is true of other number systems with integer base number, using the pure β tail, where β is the base of the number system.

However, the situation becomes much more complicated when we investigate number systems which have a non-integer base number.

EXAMPLE 1.

Let the base number $\beta = \sqrt{6} + 2 \approx 4,4495$. The digits we can use are $\mathcal{A} = \{0,1,2,3,4\}$, where the largest digit is the integer part of β. Let $\Theta = \frac{1}{\beta}$. In this

*Department of Computer Science, Széchenyi István College, Hédervári út 3., H-9026 Győr, Hungary. E-mail:kallos@rs1.szif.hu

number system $1 = 4\Theta + 2\Theta^2$, since

$$1 = 4 \cdot \frac{\sqrt{6}-2}{2} + 2 \cdot \left(\frac{\sqrt{6}-2}{2}\right)^2, \text{ with } \Theta = \frac{\sqrt{6}-2}{2}.$$

Let us substitute now in this expansion one Θ^2 with $4\Theta^3 + 2\Theta^4$. This we can do without any restriction, since we apply only usable digits. Then it derives $1 = 4\Theta + 1\Theta^2 + 4\Theta^3 + 2\Theta^4$. Repeating this method we can present infinite different expansions for 1:

$$1 = 0,42_{(\beta)} = 0,4142_{(\beta)} = 0,414142_{(\beta)} = 0,41\ldots4142_{(\beta)} = 0,4141\ldots_{(\beta)}.$$

Thus, in a general number system numbers can have more than two different expansions. The choice of the numbers with only one expansion seems to be difficult.

2 Expansions of numbers

The methods presented here are based on the work of Z. DARÓCZY and I. KÁTAI ([1] and [2]), with new approaches when needed. They have specified the univoque sequences and have presented a method for the computation of the Hausdorff dimension of the univoque set in the cases $1 < \beta \leq 2$, where β is the base of the number system. This is our goal now in general, in an arbitrary number system with base number $\beta > 1$.

As in the example let $\Theta = \frac{1}{\beta}$, and $\mathcal{A} = \{0, 1, \ldots, [\beta]\}$ the set of the usable digits. The set of fractions in this number system is

$$\mathcal{F} = \{x | x = \sum_{n=1}^{\infty} \frac{a_n}{\beta^n} = \sum_{n=1}^{\infty} a_n \Theta^n\},$$

where $a = (a_1, a_2, \ldots) \in \{0, 1, \ldots, [\beta]\}^{\mathbf{N}}$. The smallest element in this set is 0 (with all of the a_i-s $= 0$), and the largest element is

$$L = [\beta]\Theta + [\beta]\Theta^2 + [\beta]\Theta^3 + \ldots = \frac{[\beta]\Theta}{1-\Theta},$$

(with all of the a_i-s $= [\beta]$). Here $L \geq 1$, because from this inequality - substituting L - follows $[\beta] \geq \beta - 1$.

From now on we work only on the set of fractions.

For an arbitrary $x \in [0, L]$ we are able to describe at least one sequence $a = (a_1, a_2, \ldots) \in \{0, 1, \ldots, [\beta]\}^{\mathbf{N}}$, which produces the number x, i.e. $x = \sum_{n=1}^{\infty} a_n \Theta^n$. This we can do for example with the β or regular expansion of x. A. RÉNYI has proved [9], that every number x has a β expansion with $\beta > 1$ as follows:

$$x = \varepsilon_0(x) + \frac{\varepsilon_1(x)}{\beta} + \frac{\varepsilon_2(x)}{\beta^2} + \ldots,$$

where $\varepsilon_0(x) = [x], \varepsilon_1(x) = [\beta(x)], \varepsilon_2(x) = [\beta(\beta(x))], \ldots$ - here $[x]$ denotes the integral part and (x) the fractional part of x. The set $\mathcal{F}$ is closed and bounded (G. A. EDGAR [3]). For our investigation we will use two different expansions, the regular and the quasiregular one. The first is the "restriction" of the β expansion to the set of fractions.

The regular expansion. Let us define the following sequence $\varepsilon_n(x)$ for $x \in [0, L]$, by induction on n:

$\varepsilon_n(x) = j$, if

$$\sum_{i=1}^{n-1} \varepsilon_i(x)\Theta^i + j\Theta^n \leq x$$

where $j \in \mathcal{A}$, but

$$\sum_{i=1}^{n-1} \varepsilon_i(x)\Theta^i + (j+1)\Theta^n > x,$$

or $j+1 > [\beta]$, i.e. we would use a non-usable digit for the expansion. The expansion $x = \varepsilon_1(x)\Theta + \varepsilon_2(x)\Theta^2 + \ldots$ is called the regular expansion of x.

This essentially means, that we choose the largest usable digit in every step. So the expansion $1 = 0,42_{(\beta)}$ is the regular expansion of 1 in the previous example, since $4\Theta < 1$, thus $\varepsilon_1(x) = 4$, and $4\Theta + 2\Theta^2 = 1$, thus $\varepsilon_2(x) = 2$.

The quasiregular expansion. Let us define by induction on n the following sequence $\delta_n(x)$ for $x \in (0, L]$:

$\delta_n(x) = j$, if

$$\sum_{i=1}^{n-1} \delta_i(x)\Theta^i + j\Theta^n < x$$

where $j \in \mathcal{A}$, but

$$\sum_{i=1}^{n-1} \delta_i(x)\Theta^i + (j+1)\Theta^n \geq x,$$

or $j+1 > [\beta]$, i.e. we would use a non-usable digit for the expansion. The expansion $x = \delta_1(x)\Theta + \delta_2(x)\Theta^2 + \ldots$ is called the quasiregular expansion of x.

The quasiregular expansion is always infinite, and if the regular expansion is infinite too, then the two expansions are the same. Comparing with the regular expansion here we choose "almost" the largest usable digit in every step. In the previous example the quasiregular expansion of 1 is $1 = 0,4141\ldots_{(\beta)}$. Here $4\Theta < 1$ so $\delta_1(x) = 4$, $4\Theta + \Theta^2 < 1$ and $4\Theta + 2\Theta^2 = 1$, so $\delta_2(x) = 1$. $4\Theta + \Theta^2 + 4\Theta^3 < 1$, thus $\delta_3(x) = 4$, and eventually $1 = 0,4141\ldots_{(\beta)}$ follows.

In the example we have seen too, that in spite of the "closeness" of the regular and quasiregular expansions, we can find other different infinite expansions among them.

3 Univoque sequences

We call the sequence $\varepsilon \in \{0,1,\ldots,[\beta]\}^{\mathbf{N}}$ univoque (with respect to Θ) if the equation

$$\sum_{n=1}^{\infty} \varepsilon_n \Theta^n = \sum_{n=1}^{\infty} \delta_n \Theta^n$$

is only true in the case $\varepsilon = \delta$, i.e. $\varepsilon_n = \delta_n$, for $n \in \mathbf{N}$ ($\delta \in \{0,1,\ldots,[\beta]\}^{\mathbf{N}}$). In this case the number $\sum_{n=1}^{\infty} \varepsilon_n \Theta^n$ is said to be univoque, too.

The sequences

$$\underline{0} := (0,0,\ldots), \qquad \underline{[\beta]} := ([\beta],[\beta],\ldots)$$

are univoque, because every other sequence is clearly larger or smaller than these ones, respectively, using lexicographic ordering.

For $\varepsilon \in \{0,1,\ldots,[\beta]\}^{\mathbf{N}}$ let $\overline{\varepsilon} = \underline{[\beta]} - \varepsilon = ([\beta]-\varepsilon_1, [\beta]-\varepsilon_2, \ldots)$, which we will call the complementary sequence. From this it follows that $\overline{\varepsilon} \in \{0,1,\ldots,[\beta]\}^{\mathbf{N}}$, and if $x = \sum_{n=1}^{\infty} \varepsilon_n \Theta^n$, then $\sum_{n=1}^{\infty} \overline{\varepsilon_n} \Theta^n = ([\beta]-\varepsilon_1)\Theta + ([\beta]-\varepsilon_2)\Theta^2 + \ldots = L - x$.

EXAMPLE 1. (CONTINUED)

Are the following sequences univoque in the number system with base number $\beta = \sqrt{6}+2$?

a) The sequence $(3,\ldots,3,3,4,4,\ldots)$.

Here we can substitute the last digit 3 with digit 4, using for example $1 = 0,42_{(\beta)}$. Thus, if we "catch" a digit 4 and a digit 2 from the tail of the sequence, we get $(3,\ldots,3,4,0,2,4,4,\ldots)$, which represents the same number, so it is clearly not univoque. Of course, using other expansions of 1, we get infinitely many different sequences representing the same number: from $1 = 0,4142_{(\beta)}$ it derives $(3,\ldots,3,4,0,3,0,2,4,4,\ldots)$, from $1 = 0,414142_{(\beta)}$ we get $(3,\ldots,3,4,0,3,0,3,0,2,4,4,\ldots)$ etc.

We can work similarly with the complementary sequence $(1,\ldots,1,0,\ldots)$. Here we can substitute the last digit 1 with digit 0, using $1 = 0,42_{(\beta)}$. Thus we get $(1,\ldots,1,0,4,2,0,0,\ldots)$, which represents the same number, and we can produce infinitely many such sequences in the same manner.

b) The sequence $(4,\ldots,4,3,3,\ldots)$.

In this sequence we are not able to change any digit using the equation $1 = 4\Theta + 2\Theta^2$, since we would get in all cases non-usable digits: $(4,\ldots,4,2,3{+}4,3{+}2,3,3,\ldots)$ or $(4,\ldots,4,4,3{-}4,3{-}2,3,3,\ldots)$. The same is true of the complementary sequence, so we conclude, that both of the sequences are univoque.

We have seen, that the expansions of 1 play a very important role in deciding the univoque property. Since $[\beta]\Theta \leq 1$ both in the regular and the quasiregular expansion of 1 surely $\varepsilon_1 = [\beta]$ and $\delta_1 = [\beta]$ if $[\beta] < \beta$.

Now we present the exact theoretical investigation. Some of the results were already presented in [4], these parts we quote briefly, without proofs (Lemma 1., Propositon 1., Theorem 1. and 2.).

By using the regular expansions we are able to decide whether a sequence is univoque or not:

LEMMA 1. *$\varepsilon \in \{0,1,\ldots,[\beta]\}^{\mathbf{N}}$ is univoque with respect to $\Theta \iff \varepsilon$ and $[\underline{\beta}]-\varepsilon$ are regular.*

(This Lemma is a generalization of Theorem 2.1 in [1].)

By Lemma 1., in order to decide whether a sequence is univoque or not we need the regular expansion of the sequence and that of the complementary sequence. To establish that an expansion producing a number less than 1 is regular, we use the result of W. PARRY [8]. By reformulating his results according to our notations we get the following

PARRY CONDITION.

$$b = (b_1, b_2, \ldots) \text{ is regular representing a number less than } 1 \iff$$

$$(b_n, b_{n+1}, \ldots) < (\ell_1, \ell_2, \ldots) \ \forall \ n \geq 1,$$

where $0 \leq x = b_1\Theta + b_2\Theta^2 + \ldots < 1$ with $b \in \{0,1,\ldots,[\beta]\}^{\mathbf{N}}$, and $(\ell_1, \ell_2, \ldots)$ is the coefficient sequence of the quasiregular expansion of 1.

Using only the Parry condition we are not able to decide the univoque or regular property of a sequence representing a number in the interval $[1, L]$. To accomplish this we shall use other Propositions [4], and eventually it follows, that the further regular sequences are in the form $([\beta], \ldots, [\beta], b_1, b_2, \ldots)$ where the sequence $(b_1, b_2, \ldots)$ is regular representing a number less than 1.

The set of the univoque numbers. Let

$$H = \{x = \sum_{n=1}^{\infty} \varepsilon_n \Theta^n | x \in [0, L] \text{ and } \varepsilon \text{ univoque with respect to } \Theta\}$$

be the set of the univoque numbers of the interval $[0, L]$, and similarly H^* and H_1 the set of the univoque numbers of the intervals $[\Theta, 1)$, $[0, 1)$ respectively.

PROPOSITION 1. *We have*

$$H_1 = \{0\} \cup \bigcup_{n=0}^{\infty} \Theta^n H^*.$$

By Lemma 1., the location of the univoque numbers in the interval $[0, L]$ is symmetrical, and from Proposition 1., it is self-similar. Thus, if $L < 2$, then from the univoque numbers of the interval $[0, 1)$ by reflection we can get the univoque numbers in $[1, L]$, and so eventually the univoque numbers of the whole interval $[0, L]$.

$L = \frac{[\beta]\Theta}{1-\Theta} < \frac{1}{1-\Theta}$, and the fraction on the right side is less than 2 if $1-\Theta \geq \frac{1}{2}$, i.e. if $\beta \geq 2$. This will be assumed in the sequel, since the properties of the univoque set in the cases $1 < \beta \leq 2$ are already well-known ([1],[2]).

Thus, we can specify all univoque numbers (the set H) if we know the univoque numbers in the interval $[\Theta, 1)$, i.e. the set H^*.

Breaking down the problem into two cases. Let us notice now that

$$\frac{1}{K+1} < \Theta \leq \frac{1}{K}, \text{ i.e. } K = [\beta].$$

Clearly, in this interval there exists a Θ_K for which $K\Theta_K + {\Theta_K}^2 = 1$, since for all Θ in this interval $K\Theta \leq 1$ but $(K+1)\Theta > 1$. The value of this number is

$$\Theta_K = \frac{-K + \sqrt{K^2+4}}{2},$$

and if we use the notation $\beta_K = \frac{1}{\Theta_K}$, then $\beta_K = K + \Theta_K$.

The case when the fraction part is larger than Θ_K $(K+1 > \beta > K + \Theta_K)$ will be called from now on the "big case", and the case when the fraction part is smaller than Θ_K $(K \leq \beta \leq K + \Theta_K)$ the "small case".

4 Univoque sequences in the small case

Let

$$Z = \{z = \varepsilon_1\Theta + \varepsilon_2\Theta^2 + \varepsilon_3\Theta^3 + \ldots | 1 \leq \varepsilon_i \leq K-1\}.$$

THEOREM 1. *All elements of Z are univoque numbers.*

However, there are also other univoque sequences. According to our former investigation, the univoque sequences are the following:

a) The sequences $\underline{[\beta]} = \underline{K}$ and $\underline{0}$,

b) the sequences of type $[\beta]\ldots[\beta]b_ib_{i+1}\ldots$ and $0\ldots0b_ib_{i+1}\ldots$, where for the tail $b = b_ib_{i+1}\ldots$

$$\bar{t}_1\bar{t}_2\ldots < \sigma^j(b) < t_1t_2\ldots$$

is true of all $j = 0, 1, \ldots$ (σ is the shift operator).

So we can represent the whole univoque set from Z as follows:

$$H = \{0\} \cup \{L\} \cup Z \cup \bigcup_{j=1}^{\infty} \Theta^j Z \cup \bigcup_{j=1}^{\infty} (K\Theta + K\Theta^2 + \ldots + K\Theta^j + \Theta^j Z).$$

The Hausdorff dimension of the set H. The Hausdorff dimensions of the sets $\Theta^j Z$ and $K\Theta + \ldots + K\Theta^j + \Theta^j Z$ $(j = 1, 2, \ldots)$ are clearly the same as the dimension of the set Z. Thus, the Hausdorff dimension of the whole set H equals the dimension of the base set Z.

To compute the dimension of the set Z, we first specify the self-similarity dimension, and after this we check the fulfilment of the open set condition, which guarantees that the Hausdorff dimension equals the self-similarity dimension, according to the method presented by G. A. EDGAR [3].

THEOREM 2. *The Hausdorff dimension of the univoque set is*

$$\dim H = \frac{\log(K-1)}{\log\beta}.$$

REMARK. To represent the univoque sequences we can use a graphic model. We build a directed graph, the nodes of which are the usable digits in the number system, and draw an edge from the node a to b if the digit b is allowable (in a univoque sequence) after digit a. We label the edges by Θ. Thus, we get a directed graph called Mauldin-Williams graph [3]. Wandering over all the digits of the graph, we can construct all univoque sequences. The fulfilment of the open set condition guarantees, that the self-similarity dimension of the graph is the same as the Hausdorff dimension of the set H. The graph model is a useful means to demonstrate the univoque sequences, but it is not absolutely necessary.

The number system with base number $1+\sqrt{2}$.
In this number system $[\beta] = 2$, $\mathcal{A} = \{0,1,2\}$, $\Theta = \sqrt{2}-1$,

$$L = \frac{2\Theta}{1-\Theta} = \frac{2\sqrt{2}-2}{2-\sqrt{2}} = \sqrt{2}.$$

Since $2\Theta + \Theta^2 = 1$, the sequences belonging to the regular and the quasiregular expansion of 1 are 21 and $(20)^\infty$, respectively. This number system belongs to the small case, thus the univoque sequences are the following:

a) The sequences $\underline{2}$ and $\underline{0}$,

b) the sequences of type $2\ldots211\ldots$ and $0\ldots011\ldots$.

Let us denote the set of the univoque numbers beginning with i with H_i, where $i = 0,1,2$. The set of all univoque numbers is $H = H_0 \cup H_1 \cup H_2$, with

$$H_0 = (0 + \Theta H_0) \cup (0 + \Theta H_1),$$

$$H_1 = (\Theta + \Theta H_1),$$

$$H_2 = (2\Theta + \Theta H_1) \cup (2\Theta + \Theta H_2).$$

The structure of the univoque set is representable by the Mauldin-Williams graph shown on Figure 1.

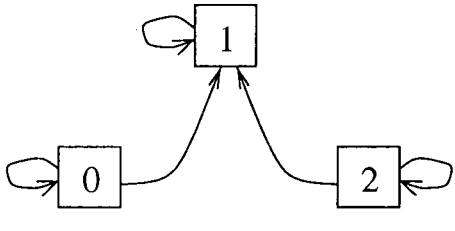


Figure 1: The Mauldin-Williams graph in the number system $1+\sqrt{2}$.

Since

$$H_1 = \{(1)^\infty\} = \{\frac{\sqrt{2}}{2}\},$$

the set H contains only countably many elements. Thus, its self-similarity and Hausdorff dimension is 0, according to Theorem 2.

The set H approximately has the form [1] shown on Figure 2.

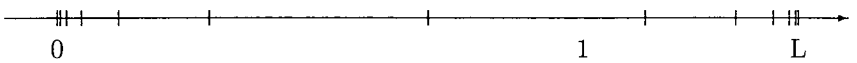


Figure 2: The approximate form of the set H

REMARK. The univoque set is very similar to this in all of the cases, when $2 < \beta \leq \beta_2 = 1 + \sqrt{2}$, since the univoque sequences have the same form. These base numbers give the simplest univoque sets investigating number systems with $\beta > 2$.

5 Univoque sequences in the big case

In the big case the structure of the univoque set is much more complicated than in the small case. Usually, the Mauldin-Williams graph of the set is not strongly connected, it contains more strongly connected parts. We build up this graph in the same manner, as before (using the Parry condition). The nodes of it represent the allowable digits (or sequence parts) in the univoque sequences (because of the complicated structure it is possible, that in the representation we have to use sequence parts - see the following example). From the node $i_1 i_2 \dots i_n$ there is an edge to the node $j_1 j_2 \dots j_n$, if $i_2 i_3 \dots i_n = j_1 j_2 \dots j_{n-1}$, and the sequence part $i_1 i_2 \dots i_n j_n$ is allowed in a univoque sequence.

THEOREM 3. *The Hausdorff dimension of the univoque set in the big case is the same, as the largest self similarity dimension of the strongly connected parts.*

PROOF. Let us assume, that the graph representing the univoque set has m strongly connected parts, $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \dots, \mathcal{D}^{(m)}$. The graph part $\mathcal{D}^{(1)}$ has a unique self-similarity dimension, let us denote this by $\dim_s \mathcal{D}^{(1)}$.

a) First we prove, that the self-similarity dimension of $\mathcal{D}^{(1)}$ is the same, as the Hausdorff dimension of $\mathcal{D}^{(1)}$ (which we denote by $\dim \mathcal{D}^{(1)}$).

To do this we have the check the open set condition. We prove this part generally, for an arbitrary (strongly) connected graph part. Let us consider for all nodes v the set I_v on the number line, which contains the picture of the sequences beginning with v. This is a closed interval. Intervals according to different nodes can not

[1] To draw Figure 2 and Figure 4 we have used the computer algebra software Maple [7]. Maple is a registered trademark of Waterloo Maple Inc.

have an intersection. Let us assume to the contrary, that $I_{i_1 i_2 \ldots i_n} \cap I_{j_1 j_2 \ldots j_n} \neq \emptyset$. Since the nodes are different, we can find different digits in the same position: $i_k \neq j_k$. It can be assumed, that for example $i_k = j_k + 1$, since if it is not possible, then clearly larger difference can not exist, too. The smallest element in the first interval is

$$i_1 \cdot \Theta + i_2 \cdot \Theta^2 + \ldots + i_k \cdot \Theta^k + 0 \cdot \Theta^{k+1} + 0 \cdot \Theta^{k+2} + \ldots$$

and the largest in the second must be smaller than

$$i_1 \cdot \Theta + i_2 \cdot \Theta^2 + \ldots + (i_k - 1) \cdot \Theta^k + \ell_1 \cdot \Theta^{k+1} + \ell_2 \cdot \Theta^{k+2} + \ldots + \ell_p \cdot \Theta^{k+p} + \ell_1 \cdot \Theta^{k+p+1} + \ldots$$

where in the following the digits $\ell_1 \ell_2 \ldots \ell_p$ are repeated, which are the coeffients in the quasiregular expansion of 1 (this result follows from the Parry condition). Thus, the intervals can not have an intersection. Choosing open intervals "little bit larger" than these closed intervals there is still no intersection, so eventually $\dim_s \mathcal{D}^{(1)} = \dim \mathcal{D}^{(1)}$.

b) As in a) we have $\dim \mathcal{D}^{(2)} = \dim_s \mathcal{D}^{(2)}$. The set $\mathcal{D}^{(1)} \cup \mathcal{D}^{(2)}$ is situated in the graph representing the whole univoque set in such manner, that we complete the two strongly connected parts with the through leading edges and nodes. Now, using the results of R. D. MAULDIN and S. C. WILLIAMS [6], we deduce that

$$\dim(\mathcal{D}^{(1)} \cup \mathcal{D}^{(2)}) = \max\{\dim_s(\mathcal{D}^{(1)}), \dim_s(\mathcal{D}^{(2)})\}.$$

In the sequel we consider the graph part containing the sets $\mathcal{D}^{(1)}$ and $\mathcal{D}^{(2)}$ as one component, and we add the set $\mathcal{D}^{(3)}$ etc. Finally we get for the Hausdorff dimension of the whole graph $\mathcal{G}$:

$$\dim \mathcal{G} = \max(\dim_s \mathcal{D}^{(1)}, \dim_s \mathcal{D}^{(2)}, \ldots, \dim_s \mathcal{D}^{(m)}),$$

and since the open set condition is satisfied, this is the Hausdorff dimension of the set H.

□

The number system with base number $\frac{3(26+6\sqrt{33})^{\frac{1}{3}}}{(26+6\sqrt{33})^{\frac{2}{3}} - 8 - (26+6\sqrt{33})^{\frac{1}{3}}}$.

In this number system [2] $\beta \approx 3,3830$, thus $\mathcal{A} = \{0, 1, 2, 3\}$. The regular expansion of 1 is $1 = 3\Theta + \Theta^2 + \Theta^3$, and the quasiregular form is $1 = 0,310310\ldots$. Thus, the univoque sequences can not contain the following parts:

$$(3,3), (0,0), (3,2), (0,1), (3,1,3), (0,2,0), (3,1,2), (0,2,1), (3,1,1), (0,2,2).$$

The possible parts are the following:

$$(3,1,0), (3,0,3), (3,0,2), (2,3,1), (2,3,0), (2,2,3), (2,2,2), (2,2,1), (2,2,0),$$

[2] The base number of the number system is the reciprocal value of the real solution of the equation $1 = 3\Theta + \Theta^2 + \Theta^3$

$$(2,1,3),(2,1,2),(2,1,1),(2,1,0),(2,0,3),(2,0,2),$$

and their complementers:

$$(0,2,3),(0,3,0),(0,3,1),(1,0,2),(1,0,3),(1,1,0),(1,1,1),(1,1,2),(1,1,3),$$

$$(1,2,0),(1,2,1),(1,2,2),(1,2,3),(1,3,0),(1,3,1).$$

Now if we would like to denote the possible parts unambigous, then we have to use three digits in a node. Similarly as before, we will use the sets $H_{i_1 i_2 i_3}$, and we can write the set equations, for example:

$$H_{023} = (0 + \Theta H_{231}) \cup (0 + \Theta H_{230})$$

$$H_{102} = \Theta + \Theta H_{023}$$

$$H_{111} = (\Theta + \Theta H_{110}) \cup (\Theta + \Theta H_{111}) \cup (\Theta + \Theta H_{112}) \cup (\Theta + \Theta H_{113})$$

...

To save place, we have omitted the further equations, but it is an easy exercise to write those ones, too. The structure of the univoque set is representable with the Mauldin-Williams graph shown on Figure 3. The graph contains two strongly connected parts. To indicate this, we have separated the nodes. The nodes which have "through leading" role are shown alone, and their edges indicate the connections.

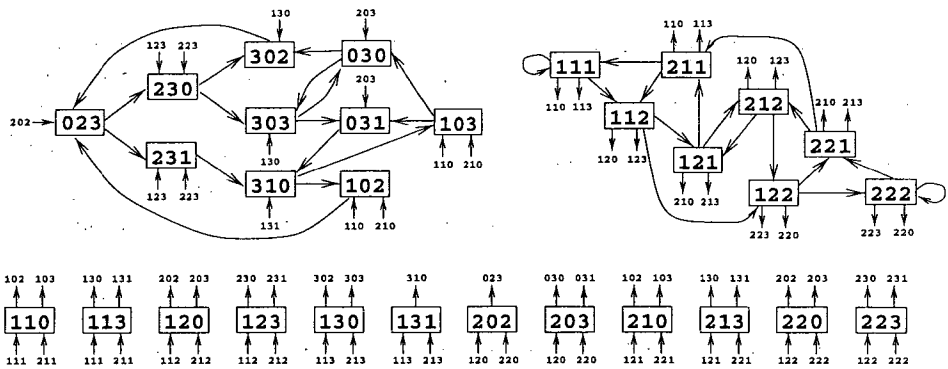


Figure 3: Mauldin-Williams graph. (See text for details.)

To specify the self-similarity dimension, for the strongly connected graph parts we get the following equation systems:

The first graph part

$$q^s_{023} = \lambda \cdot q^s_{230} + \lambda \cdot q^s_{231}$$
$$q^s_{030} = \lambda \cdot q^s_{302} + \lambda \cdot q^s_{303} \qquad = q^s_{230}$$
$$q^s_{031} = \lambda \cdot q^s_{310} \qquad = q^s_{231}$$

$$q^s_{102} = \lambda \cdot q^s_{023} \qquad = q^s_{302}$$
$$q^s_{103} = \lambda \cdot q^s_{030} + \lambda \cdot q^s_{031} \qquad = q^s_{303}$$
$$q^s_{310} = \lambda \cdot q^s_{102} + \lambda \cdot q^s_{103},$$

where q_{ijk} is the Perron number belonging to the node ijk, s is the self-similarity dimension of the graph part, and $\lambda = \Theta^s$. After repeated substitution

$$q^s_{023} = \lambda \cdot q^s_{030} + \lambda^2 \cdot q^s_{030}$$
$$q^s_{030} = \lambda^2 \cdot q^s_{023} + \lambda \cdot q^s_{023}$$

We can choose without any restriction one of the Perron numbers ([3]), let for example $q_{023} = 1$. Thus, from the last equations

$$1 = \lambda \cdot q^s_{030} + \lambda^2 \cdot q^s_{030} \text{ and}$$
$$q^s_{030} = \lambda^2 + \lambda.$$

From these two equations $1 = \lambda^2 + 2\lambda^3 + \lambda^4 = (\lambda + \lambda^2)^2$. The solution of $\lambda^2 + \lambda - 1 = 0$ is

$$\lambda = \frac{\sqrt{5} - 1}{2},$$

so the dimension is

$$s = \frac{\log(\frac{\sqrt{5}+1}{2})}{\log\beta} \approx 0,3948.$$

The second graph part

$$q^s_{111} = \lambda \cdot q^s_{111} + \lambda \cdot q^s_{112} \qquad = q^s_{211}$$
$$q^s_{112} = \lambda \cdot q^s_{121} + \lambda \cdot q^s_{122} \qquad = q^s_{212}$$
$$q^s_{121} = \lambda \cdot q^s_{211} + \lambda \cdot q^s_{212} \qquad = q^s_{221}$$
$$q^s_{122} = \lambda \cdot q^s_{221} + \lambda \cdot q^s_{222} \qquad = q^s_{222}$$

After repeated substitution

$$q^s_{111} = \lambda \cdot q^s_{111} + \lambda \cdot q^s_{112} \qquad = q^s_{112}$$

From this $q^s_{111} = 2\lambda \cdot q^s_{111}$, and $\lambda = \frac{1}{2}$.
Thus, the dimension is

$$s = \frac{\log 2}{\log\beta} \approx 0,5687,$$

and the dimension of the whole graph is eventually the dimension of the second graph part. The open set criterion is now satisfied, so this is the Hausdorff dimension of the whole univoque set, which approximately has the form shown on Figure 4.

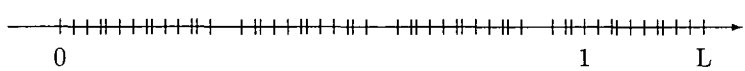


Figure 4: The approximate form of the set H

References

[1] Z. DARÓCZY, I. KÁTAI, Univoque Sequences, *Publ. Math. Debrecen*, 42. (1993), 397-407.

[2] Z. DARÓCZY, I. KÁTAI, On the Structure of Univoque Numbers, *Publ. Math. Debrecen*, 46. (1995), 385-408.

[3] G. A. EDGAR, *Measure, Topology and Fractal Geometry*, Springer Verlag, New York, 1994.

[4] G. KALLÓS, The Structure of the Univoque Set in the Small Case, *Publ. Math. Debrecen*, 53. (1998). (to appear)

[5] D. E. KNUTH, *The Art of Computer Programming, Vol. 2.*, Addison-Wesley, 1981.

[6] R. D. MAULDIN, S. C. WILLIAMS, Hausdorff Dimension in Graph Directed Constructions, *Trans. Amer. Math. Soc.*, 309. (1988), 811-819.

[7] MOLNÁRKA GYŐZŐ ET AL., *A Maple V és alkalmazásai*, Springer, 1996.

[8] W. PARRY, On the β Expansions of Real Numbers, *Acta Math. Hung.*, 11. (1960), 401-406.

[9] A. RÉNYI, Representations for Real Numbers and their Ergodic Properties, *Acta Math. Hung.*, 8. (1957), 477-493.

Acta Cybernetica 14 (1999) 315-330.

Optimal parameters of a sinusoidal representation of signals

A. Kocsor* L. Tóth* I. Bálint[†‡§]

Abstract

In the spectral analysis of digital signals, one of the most useful parametric models is the representation by a sum of phase-shifted sinusoids in form of $\sum_{n=0}^{N-1} A_n \sin(\omega_n t + \varphi_n)$, where A_n, ω_n, and φ_n are the component's amplitude, frequency and phase, respectively. This model generally fits well speech and most musical signals due to the shape of the representation functions. If using all of the above parameters, a quite difficult optimization problem arises. The applied methods are generally based on eigenvalue decomposition [3]. However this procedure is computationally expensive and works only if the sinusoids and the residual signal are statistically uncorrelated. To speed up the representation process also rather ad hoc methods occur [4]. The presented algorithm applies the newly established Homogeneous Sinus Representation Function (HSRF) to find the best representing subspace of fixed dimension N by a BFGS optimization. The optimum parameters $\{A, \omega, \varphi\}$ ensure the mean square error of approximation to be below a preset threshold.

1 Introduction

Since the invention of the telephone, speech or generally sound processing and representation have paramount importance in electrical engineering. In the last years the rapid development of multimedia and computer networks brought a revival of the high-effective coding and representation problem.

By the classic model of speech generation, the voiced part of speech comes from the oscillation of the vocal chord, which is modellable by an oscillating string. The voice consists of a fundamental and its harmonics, therefore it is well representable

*MTA-JATE Research Group on Artificial Intelligence, H-6720 Szeged, Aradi Vértanuk Tere 1, Hungary

†Department of Theoretical Physics, József Attila University, H-6720 Szeged, Tisza L. krt. 80-82., Hungary

‡Department of Pharmaceutical Analysis, Szent-Györgyi Albert Medical University, H-6720 Szeged, Somogyi Béla u. 4., Hungary

§Department of Natural Scienses, Polytechnic of the Miskolc University, H-2400 Dunaújváros, Táncsics M. u. 1., Hungary

in the form

$$\sum_{n=0}^{N-1} A_n \sin(\omega_n t + \varphi_n).$$

The error of this approximation gives the 'unvoiced', noise-like part, which can be decoupled from the signal. The model fits well also musical signals, since the sound of most musical instruments (stringed-, wind instruments, etc.) consist of harmonic sinusoids. The residual signal again contains the noise-like part of the sound (e.g. drum hits), which should be modelled separately.

The above form of the model yields a complicated optimization problem enforcing some simplifications. In case of DFT (Discrete Fourier Transform), the number of sinusoids and their frequencies are fixed providing a rapid way for the computation of amplitudes and phases. However, in general, the individual sinusoidal components of this representation may significantly differ in their parameters from the real sound components. The method of McAulay-Quatieri [4] tends to deduce the real frequencies of components by looking for peaks in the DFT spectrum. A basically different approach is based on eigenvalue decomposition [3]. Here, only the dimension of approximation space is fixed, but the statistical independence of the representation functions (sinusoids) and of the residual signal is required.

The presented procedure is free from requiring any statistical condition, only the dimension of approximation space is fixed. The established optimization problem is based on a recently introduced functional [16] and it is solved very effectively by the BFGS [10,12,13] method. The efficiency of the method is illustrated by representations of artificial and natural voice patterns.

As to the structure of the report, the second section provides the usual, 'conservative' formulation of the problem, the third section deals with the introduced Homogeneous Sinus Representation Function (HSRF), the fourth section investigates the properties of HSRF, the fifth section discusses the workhorse optimization scheme BFGS, finally the sixth section delivers the numerical illustrations and conclusions.

1.1 Notational conventions

The Euclidean norm is denoted by $\| \|$, the gradient of a function $f(x) : \mathbb{R}^n \to \mathbb{R}$ by

$$\nabla f(x) = [\frac{\partial f(x)}{\partial x_1}, \cdots, \frac{\partial f(x)}{\partial x_n}]^\top,$$

and the Hessian will be denoted, as

$$\nabla \times \nabla f(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{pmatrix}.$$

Definition 1.1 *The continuous function* $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ *is homogeneous of* l*th degree, if*

$$f(k\mathbf{x}) = k^l f(\mathbf{x}), \qquad k \in \mathbb{R}.$$

2 The representation problem

A signal is sampled at points $\tau_0, \tau_1, \cdots, \tau_{K-1}$ of a closed time-interval $[0, \tau]$ and the obtained values are represented by the real sequence,

$$x[\tau_0], \cdots, x[\tau_{K-1}].$$

A function of the form

$$\sum_{n=0}^{N-1} A_n \sin(\omega_n t + \varphi_n)$$

is sought, which approximates the measured sample with a preset error $\epsilon > 0$, by fixing the dimension of approximation space to N,

$$\min_{\substack{A_1, \cdots, A_{N-1} \\ \omega_1, \cdots, \omega_{N-1} \\ \varphi_1, \cdots, \varphi_{N-1}}} \sum_{k=0}^{K-1} \left(\sum_{n=0}^{N-1} A_n \sin(\tau_k \omega_n + \varphi_n) - x[\tau_k] \right)^2 < \epsilon. \tag{1}$$

3 Optimization of the Homogeneous Sinusoidal Representation Function

Let be introduced the following notation,

$$w_k(\mathbf{A}, \omega, \varphi) := \sum_{n=0}^{N-1} A_n \sin(\omega_n \tau_k + \varphi_n), \quad k = 0, \cdots, K-1 \tag{2}$$

where

$$\mathbf{A} := [A_0, \cdots, A_{N-1}]^\top, \quad \omega := [\omega_0, \cdots, \omega_{N-1}]^\top, \quad \varphi := [\varphi_0, \cdots, \varphi_{N-1}]^\top$$

and let be applied the trigonometric identity,

$$w_k(\mathbf{A}, \omega, \varphi) := \sum_{n=0}^{N-1} A_n \sin(\tau_k \omega_n + \varphi_n) =$$

$$\sum_{n=0}^{N-1} A_n (\sin(\omega_n \tau_k) \cos(\varphi_n) + \cos(\omega_n \tau_k) \sin(\varphi_n)) = \sum_{n=0}^{N-1} a_n \sin(\omega_n \tau_k) + b_n \cos(\omega_n \tau_k),$$

where

$$a_n = A_n \cos(\varphi_n), \quad b_n = A_n \sin(\varphi_n). \tag{3}$$

By introducing new variables,

$$\frac{c_n}{\sqrt{c_n^2 + d_n^2}} := \sin(\omega_n), \quad \frac{d_n}{\sqrt{c_n^2 + d_n^2}} := \cos(\omega_n), \qquad c_n^2 + d_n^2 \neq 0, \tag{4}$$

we obtain

$$\sum_{n=0}^{N-1} a_n \sin(\omega_n \tau_k) + b_n \cos(\omega_n \tau_k) =$$

$$\sum_{n=0}^{N-1} a_n \sin(\tau_k \arcsin(\frac{c_n}{\sqrt{c_n^2 + d_n^2}})) + b_n \cos(\tau_k \arcsin(\frac{c_n}{\sqrt{c_n^2 + d_n^2}})) =: w_k(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}),$$

where

$$\mathbf{a} := [a_0, \cdots, a_{N-1}]^\top, \ \mathbf{b} := [b_0, \cdots, b_{N-1}]^\top,$$
$$\mathbf{c} := [c_0, \cdots, c_{N-1}]^\top, \ \mathbf{d} := [d_0, \cdots, d_{N-1}]^\top.$$

If we introduce two further vectors,

$$\mathbf{x} := [x[\tau_0], \cdots, x[\tau_{K-1}]]^\top, \ \mathbf{w} := [w_0(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}), \cdots, w_{K-1}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})]^\top$$

the Homogeneous Sinusoidal Representation Function (HSRF) to be optimized will be

$$L_{\mathbf{xw}}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}) := \mathbf{x}^\top \mathbf{x} \mathbf{w}^\top \mathbf{w} - (\mathbf{x}^\top \mathbf{w})^2. \tag{5}$$

4 Some properties of HSRF

Notation 4.1 *Let the parameters* $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ *of HSRF be concatenated into a single vector, as follows*

$$\mathbf{z} = [\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}]^\top = [a_0, \cdots, a_{N-1}, b_0, \cdots, b_{N-1}, c_0, \cdots, c_{N-1}, d_0, \cdots, d_{N-1}]^\top.$$

Vector $\mathbf{z}$ is of $4N$-dimension and the concatenated components occupy the following fields:

$$z_i = \begin{cases} a_i & ,if \quad 0 \leq i \leq N-1 \\ b_{i-N} & ,if \quad N \leq i \leq 2N-1 \\ c_{i-2N} & ,if \quad 2N \leq i \leq 3N-1 \\ d_{i-3N} & ,if \quad 3N \leq i \leq 4N-1 \end{cases}$$

Lemma 4.2 *HSRF exhibits the properties:*

1. $L_{\mathbf{xw}}(\mathbf{z})$ *is a homogeneous function of 2nd degree.*

2. $L_{\mathbf{xw}}$ *is a 0th degree homogeneous function of its variables* $\mathbf{c}, \mathbf{d}$*:*

$$L_{\mathbf{xw}}(\mathbf{a}, \mathbf{b}, \lambda\mathbf{c}, \lambda\mathbf{d}) = L_{\mathbf{xw}}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}), \quad 0 \neq \lambda \in \mathbb{R}$$

3. $L_{\mathbf{xw}}$ *is a 2nd degree homogeneous function of its variables* $\mathbf{a}, \mathbf{b}$*:*

$$L_{\mathbf{xw}}(\lambda\mathbf{a}, \lambda\mathbf{b}, \mathbf{c}, \mathbf{d}) = \lambda^2 L_{\mathbf{xw}}(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}), \quad \lambda \in \mathbb{R}$$

4. $L_{\mathbf{xw}}(\mathbf{z}) = L_{\mathbf{wx}}(\mathbf{z})$

Proof. Point 4. satisfies trivially, points 1., 2., and 3. follow from the continuity of $L_{\mathbf{xw}}(\mathbf{z})$, as well as from the enumerated properties obeyed by $w_k(\mathbf{z}) \equiv w_k(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$:

1. $w_k(\lambda\mathbf{z}) = \lambda w_k(\mathbf{z}), \qquad 0 \neq \lambda \in \mathbb{R}.$
2. $w_k(\lambda\mathbf{a}, \lambda\mathbf{b}, \mathbf{c}, \mathbf{d}) = \lambda w_k(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}), \qquad \lambda \in \mathbb{R}.$
3. $w_k(\mathbf{a}, \mathbf{b}, \lambda\mathbf{c}, \lambda\mathbf{d}) = w_k(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}), \qquad 0 \neq \lambda \in \mathbb{R}.$ □

Theorem 4.3 *HSRF exhibits the enumerated properties:*

1. $L_{\mathbf{xw}}(\mathbf{z}) \geq 0$ *and* $L_{\mathbf{xw}}(\mathbf{z}) = 0$ *if and only if the* $\mathbf{x}$ *and* $\mathbf{w}$ *vectors are linearly dependent.*
2. *If* $\mathbf{z}$ *is an optimumpoint of* $L_{\mathbf{xw}}(\mathbf{z})$*, then* $L_{\mathbf{xw}}(\mathbf{z}) = 0$.
3. *If* $L_{\mathbf{xw}}(\mathbf{z}) = 0$*, then* $\nabla L_{\mathbf{xw}}(\mathbf{z}) = \mathbf{0}$.

Proof.

1. Function $L_{\mathbf{xw}}(\mathbf{z})$ stems from the Cauchy-Scwartz-Bunyakovszkij inequality applied on the vectors $\mathbf{x} \in \mathbb{R}^K$ and $\mathbf{w}(\mathbf{z}) \in \mathbb{R}^K$:

$$\mathbf{x}^\top \mathbf{x}\mathbf{w}(\mathbf{z})^\top \mathbf{w}(\mathbf{z}) \geq \left(\mathbf{x}^\top \mathbf{w}(\mathbf{z})\right)^2.$$

   The equality satisfies, if the vectors are linearly dependent,

$$\mathbf{x}^\top \mathbf{x}\mathbf{w}(\mathbf{z})^\top \mathbf{w}(\mathbf{z}) - \left(\mathbf{x}^\top \mathbf{w}(\mathbf{z})\right)^2 = 0.$$

2. If $\mathbf{z} \in \mathbb{R}^{4N}$ is an optimumpoint of $L_{\mathbf{xw}}(\mathbf{z})$, then necessarily $\nabla L_{\mathbf{xw}}(\mathbf{z}) = \mathbf{0}$. Euler's theorem ensures that the 2nd degree, homogeneous function $L_{\mathbf{xw}}(\mathbf{z})$ obeyes the equality:

$$\mathbf{z}^\top \nabla L_{\mathbf{xw}}(\mathbf{z}) = 2L_{\mathbf{xw}}(\mathbf{z}).$$

   Therefore a zerovector gradient implies a zero function value, $\nabla L_{\mathbf{xw}}(\mathbf{z}) = \mathbf{0} \Longrightarrow L_{\mathbf{xw}}(\mathbf{z}) = 0$.

3. If $L_{\mathbf{xw}}(\mathbf{z}) = 0$, the vectors $\mathbf{x}$ and $\mathbf{w}$ are linearly dependent, i.e. $\mathbf{x} = \lambda \mathbf{w}(\mathbf{z})$ without restricting generality. The following sequence of equalities proves the statement:

$$\nabla L_{\mathbf{xw}}(\mathbf{z}) =$$
$$= \mathbf{x}^\top \mathbf{x} \nabla \left(\mathbf{w}(\mathbf{z})^\top \mathbf{w}(\mathbf{z})\right) + \nabla \left(\mathbf{x}^\top \mathbf{x}\right) \mathbf{w}(\mathbf{z})^\top \mathbf{w}(\mathbf{z}) - 2 \left(\mathbf{x}^\top \mathbf{w}(\mathbf{z})\right) \nabla \left(\mathbf{x}^\top \mathbf{w}(\mathbf{z})\right) =$$
$$= \lambda^2 \left(\mathbf{w}(\mathbf{z})^\top \mathbf{w}(\mathbf{z})\right) \mathbf{w}(\mathbf{z})^\top \nabla \left(\mathbf{w}(\mathbf{z})\right) + 0 - 2\lambda^2 \left(\mathbf{w}(\mathbf{z})^\top \mathbf{w}(\mathbf{z})\right) \mathbf{w}(\mathbf{z})^\top \nabla \left(\mathbf{w}(\mathbf{z})\right) = 0$$

□

The properties of $L_{\mathbf{xw}}(\mathbf{z})$ discussed above ensure good optimization properties. The optimumpoints of this non-negative homogeneous function are global and a gradient-based optimization scheme may efficiently localize them.

Lemma 4.4 *The gradient* $\nabla L_{\mathbf{xw}}(\mathbf{z})$; $\mathbf{z} = [\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}]^\top$ *is of the form:*

$$\frac{\partial}{\partial z_i} L_{\mathbf{xw}}(\mathbf{z}) = \left(\sum_{k=0}^{K-1} x[\tau_k]^2\right) \left(2 \sum_{k=0}^{K-1} w_k(\mathbf{z}) \frac{\partial}{\partial z_i} w_k(\mathbf{z})\right) - 2 \left(\sum_{k=0}^{K-1} x[\tau_k] w_k(\mathbf{z})\right) \left(\sum_{k=0}^{K-1} x[\tau_k] \frac{\partial}{\partial z_i} w_k(\mathbf{z})\right).$$

The partial derivatives by the various sets of variables are as follow:

$$\frac{\partial}{\partial a_i} w_k(\mathbf{z}) = \sin\left(\tau_k \arcsin\left(\frac{c_i}{\sqrt{c_i^2 + d_i^2}}\right)\right)$$

$$\frac{\partial}{\partial b_i} w_k(\mathbf{z}) = \cos\left(\tau_k \arcsin\left(\frac{c_i}{\sqrt{c_i^2 + d_i^2}}\right)\right)$$

$$\frac{\partial}{\partial c_i} w_k(\mathbf{z}) = \frac{\tau_k d_i \left(a_i \cos\left(\tau_k \arcsin\left(\frac{c_i}{\sqrt{c_i^2+d_i^2}}\right)\right) - b_i \sin\left(\tau_k \arcsin\left(\frac{c_i}{\sqrt{c_i^2+d_i^2}}\right)\right)\right)}{(c_i^2 + d_i^2)}$$

$$\frac{\partial}{\partial d_i} w_k(\mathbf{z}) = -\frac{\tau_k c_i \left(a_i \cos\left(\tau_k \arcsin\left(\frac{c_i}{\sqrt{c_i^2+d_i^2}}\right)\right) - b_i \sin\left(\tau_k \arcsin\left(\frac{c_i}{\sqrt{c_i^2+d_i^2}}\right)\right)\right)}{(c_i^2 + d_i^2)}$$

Proof. The proof is trivial by the differentiation rules. □

The next theorem provides the bridge between the function value of HSRF and the representation problem (1).

Theorem 4.5 *If for any positive number* δ *and for real vectors* $\mathbf{z} \in \mathbb{R}^{4N}$ *and* $\mathbf{x} \in \mathbb{R}^K$, $L_{\mathbf{xw}}(\mathbf{z}) < \delta$ *is satisfied, then*

$$\min_i \left\| \mathbf{x} - \frac{\mathbf{w}(\mathbf{z})}{\lambda_i} \right\|^2 < \frac{2\delta}{\|\mathbf{w}(\mathbf{z})\|^2}, \qquad \lambda_i = \frac{w_i(\mathbf{z})}{x[\tau_i]}, \; i \in \{0, \cdots K-1\}.$$

Proof. For the sake of simplicity, the argument $\mathbf{z}$ of $\mathbf{w}(\mathbf{z})$ and of $w_i(\mathbf{z})$ will be omitted and $x[\tau_i]$ will be denoted simply as x_i.

$$L_{\mathbf{xw}}(\mathbf{z}) = \mathbf{x}^\top \mathbf{x}\mathbf{w}^\top \mathbf{w} - (\mathbf{x}^\top \mathbf{w})^2 = \left(\sum_{i=0}^{K-1} x_i^2\right)\left(\sum_{i=0}^{K-1} w_i^2\right) - \left(\sum_{i=0}^{K-1} x_i w_i\right)^2 =$$

$$= \frac{1}{2}\sum_{i=0}^{K-1}\sum_{j=0}^{K-1}(w_i x_j - w_j x_i)^2 = \frac{1}{2}\sum_{i=0}^{K-1}\sum_{j=0}^{K-1}\left(w_i x_j - w_j \frac{w_i}{\lambda_i}\right)^2 =$$

$$= \frac{1}{2}\sum_{i=0}^{K-1} w_i^2 \sum_{j=0}^{K-1}\left(x_j - \frac{w_j}{\lambda_i}\right)^2 = \frac{1}{2}\sum_{i=0}^{K-1} w_i^2 \left\|\mathbf{x} - \frac{\mathbf{w}}{\lambda_i}\right\|^2 < \delta$$

If choosing the 'best' of λ_is, the inequality

$$\frac{1}{2}\|\mathbf{w}\|^2 \min_i \left\|\mathbf{x} - \frac{\mathbf{w}}{\lambda_i}\right\|^2 < \frac{1}{2}\sum_{i=0}^{K-1} w_i^2 \left\|\mathbf{x} - \frac{\mathbf{w}}{\lambda_i}\right\|^2 < \delta,$$

proves the statement. □

If the previous optimization yields a $\mathbf{z_0}$ satisfying

$$\left\|\mathbf{x} - \mathbf{w}(\mathbf{z_0})\frac{x[\tau_s]}{w_s(\mathbf{z})}\right\|^2 = \min_i \left\|\mathbf{x} - \mathbf{w}(\mathbf{z_0})\frac{x[\tau_i]}{w_i(\mathbf{z})}\right\|^2 < \frac{2L_{\mathbf{xw}}(\mathbf{z_0})}{\|\mathbf{w}(\mathbf{z_0})\|^2},$$

the difference of the Euclidean norm of the signal vector $\mathbf{x}$ and the representation vector

$$\mathbf{w}(\mathbf{z_0})\frac{x[\tau_s]}{w_s(\mathbf{z})}$$

is given by the above expression.

5 Solving the representation problem by NHSRF

5.1 Application of the BFGS optimization scheme

For minimizing the representation functional the most suitable procedure proved to be the gradient based Broyden-Fletcher-Goldfarb-Shanno (BFGS) scheme [10,12,13]. Since also the zero-vector is an optimumpoint of $L_{\mathbf{xw}}(\mathbf{z})$ to avoid convergence to the zero-vector, the HSRF is normalized to be a 0th degree homogeneous function of the form,

$$\frac{L_{\mathbf{xw}}(\mathbf{z})}{\|\mathbf{a}\|\,\|\mathbf{b}\|}. \tag{6}$$

This will be called Normalized HSRF, NHSRF in short. Every former obtained result are inherited by NHSRF, however the new partial derivative components of the gradient are given below,

$$\frac{\partial}{\partial a_i}\frac{L_{\mathbf{xw}}(\mathbf{z})}{\|\mathbf{a}\|\,\|\mathbf{b}\|} = \frac{1}{\|\mathbf{b}\|}\frac{\|\mathbf{a}\|\left(\frac{\partial}{\partial a_i}L_{\mathbf{xw}}(\mathbf{z})\right) - L_{\mathbf{xw}}(\mathbf{z})\frac{a_i}{\|\mathbf{a}\|}}{\|\mathbf{a}\|^2},$$

$$\frac{\partial}{\partial b_i}\frac{L_{\mathbf{xw}}(\mathbf{z})}{\|\mathbf{a}\|\,\|\mathbf{b}\|} = \frac{1}{\|\mathbf{a}\|}\frac{\|\mathbf{b}\|\left(\frac{\partial}{\partial b_i}L_{\mathbf{xw}}(\mathbf{z})\right) - L_{\mathbf{xw}}(\mathbf{z})\frac{b_i}{\|\mathbf{b}\|}}{\|\mathbf{b}\|^2},$$

$$\frac{\partial}{\partial c_i}\frac{L_{\mathbf{xw}}(\mathbf{z})}{\|\mathbf{a}\|\,\|\mathbf{b}\|} = \frac{1}{\|\mathbf{a}\|\,\|\mathbf{b}\|}\frac{\partial}{\partial c_i}L_{\mathbf{xw}}(\mathbf{z}),$$

$$\frac{\partial}{\partial d_i}\frac{L_{\mathbf{xw}}(\mathbf{z})}{\|\mathbf{a}\|\,\|\mathbf{b}\|} = \frac{1}{\|\mathbf{a}\|\,\|\mathbf{b}\|}\frac{\partial}{\partial d_i}L_{\mathbf{xw}}(\mathbf{z}).$$

Every test result displayed for illustration was obtained by the NHSRF. For terminating the line-search

$$\min_{\kappa\in\Re}\frac{L_{\mathbf{xw}}(\mathbf{z}+\kappa\mathbf{d})}{\|\mathbf{a}\|\,\|\mathbf{b}\|},$$

the Wolf-condition, for initializing the H-matrix, the unit matrix was used. For terminating the whole BFGS optimization generally the acceptable low norm of the error-vector, as well as that of the gradient was used. We also have stopped the iteration, if the condition

$$\frac{2L_{\mathbf{xw}}(\mathbf{z})}{\|\mathbf{w}(\mathbf{z})\|^2} < \delta$$

as referred in Theorem 4.5 was satisfied.

5.2 Estimation of the number of necessary operations

For one iteration step of the BFGS scheme generally the function value and gradient should be computed at several points in the line-search process. This requires to evaluate scalar products, which can be obtained with $o(NK + K^2)$ operations, because for any k,

$$\sum_{n=0}^{N-1} a_n \sin(\tau_k \arcsin(\frac{c_n}{\sqrt{c_n^2+d_n^2}})) + b_n \cos(\tau_k \arcsin(\frac{c_n}{\sqrt{c_n^2+d_n^2}})) = w_k(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d})$$

can be obtained with $o(N)$ operations, and the K-dimensional vector $\mathbf{w}$ is obtainable with $o(NK)$ operations. To evaluate the function value, as well as the gradient, scalar products of K-dimensional vectors have to be computed. The update of the $4N\times 4N$-size H-matrix requires $o(N^2)$ operations in every step counting altogether $o(N^2+NK)$ operations per step. Since generally the number N of sinusoidal components is much less then the number of the components in the signal vector, the number of really required operations is of $o(NK)$.

6 Numerical illustration of the algorithm

The aim is to construct an acceptably accurate sinusoidal representation of an arbitrary (sound) signal. The procedure is based on the optimization of NHSRF starting from an approximate sinusoidal decomposition of the signal. The procedure

aims to reduce the number of sinusoidal components by retaining the accuracy of the representation. The numerical illustrations are mainly artificial examples to obtain a well-defined measure for the accuracy of approximation, however the representation of a natural sound sample is also included.

In all cases of artificial and natural sound patterns an approximate sinusoidal represenation served as initial parameters of the NHRSF optimization. The $x[\tau_0], \cdots, x[\tau_{K-1}]$ signal was decomposed by the following iterative algorithm:

- In every iteration, first the DFT of the signal was computed on a zero padded 2048 base point data set using a Hamming window. The obtained spectrum is the convolution of the transformed signal and the transformed window.
- The maximum amplitude component was selected. To remove the unpleasant effect of windowing, the Fourier-transform of the window function was subtracted from the spectrum after a suitable shifting and scaling.
- The iteration was continued until the largest amplitude was smaller than a preset positive number.

The iterations steps necessary for the NHSRF optimization algorithm to reach the required accuracy was empirically tested. Both in the case of the natural and artificial tests the number of function and gradient evaluations necessary for one iteration step was generally one or two.

6.1 Representation of artificial signals

The sample to be represented was a sum of N sinusoidal components sampled at K points, where the parameters $A_0, \cdots, A_{N-1}$; $\omega_0, \cdots, \omega_{N-1}$; and $\varphi_0, \cdots, \varphi_{N-1}$ were specified:

$$x[k] = \sum_{n=0}^{N-1} A_n \sin(\omega_n k + \varphi_n), \quad k \in \{1, \cdots, K\}.$$

Using the above mentioned DFT-based decomposition of the signal,

$$x[1], \cdots, x[K],$$

the following estimate was obtained,

$$x[k] \approx \sum_{n=0}^{P-1} A'_n \sin(\omega'_n k + \varphi'_n), \quad k \in \{1, \cdots, K\}, \quad , P > N,$$

which proved to be generally unacceptably inaccurate. Without restricting generality, we can assume the following ordering of the components $A'_i \geq A'_j \Leftrightarrow i \geq j$; which selects the N components,

$$A'_0, \cdots, A'_{N-1}, \omega'_0, \cdots, \omega'_{N-1}, \varphi'_0, \cdots, \varphi'_{N-1}$$

ω	φ	A	K	iter	δ_1	δ_2
0.2	0.1	2	100	8	0.0006	10^{-17}
0.2	0.1	2	200	9	0.0031	10^{-19}
0.2	0.1	2	400	7	0.0131	10^{-15}

Table 1: Approximation of one sinusoidal component.

dominating by amplitude. The optimization process started in every case from these dominant components by constructing the initial parameter vector

$$\mathbf{z} = [a'_0, \cdots, a'_{N-1}, b'_0, \cdots, b'_{N-1}, c'_0, \cdots, c'_{N-1}, d'_0, \cdots, d'_{N-1}]^\top$$

using (3) and (4). Let be assumed, that the optimization of (6) resulted in the optimum vector

$$\mathbf{z_0} = [a''_0, \cdots, a''_{N-1}, b''_0, \cdots, b''_{N-1}, c''_0, \cdots, c''_{N-1}, d''_0, \cdots, d''_{N-1}]^\top$$

and the inverses of transformations (3) and (4) yielded the parameters

$$A''_0, \cdots, A''_{N-1}, \omega''_0, \cdots, \omega''_{N-1}, \varphi''_0, \cdots, \varphi''_{N-1}.$$

The error of the DFT-based signal representation is

$$\delta_1 = \sum_{n=0}^{N-1} (A_n - A'_n)^2 + (\omega_n - \omega'_n)^2 + (\varphi_n - \varphi'_n)^2, \tag{7}$$

while that of the NHSRF-based signal representation is

$$\delta_2 = \sum_{n=0}^{N-1} (A_n - A''_n)^2 + (\omega_n - \omega''_n)^2 + (\varphi_n - \varphi''_n)^2. \tag{8}$$

For $N = 1, 2, 3$, three examples were investigated in each case and the results are displayed in Tables 1.-9.. The rows of the tables display the parameters of the sinusoidal basis functions ω, φ, A, the number of sample points K, the number of iterations *iter* and the accuracies of DFT-based and NHRSF-based representations δ_1, δ_2. The discussion of the results will be given together with the discussion of the natural test results.

6.2 Representation of natural sound signals

To check the accuracy and efficiency of the proposed algorithm on natural sound patterns, the phone /a/ was represented and synthesised by the DFT decomposition, as well as by the NHSRF optimization based scheme. The sample was consisting in 861 points from the middle of a, the DFT algorithm described above

ω	φ	A	K	iter	δ_1	δ_2
0.2	0.1	200	100	7	0.2456	10^{-15}
0.2	0.1	200	200	10	0.1858	10^{-16}
0.2	0.1	200	400	11	0.0456	10^{-15}

Table 2: Approximation of one sinusoidal component.

ω	φ	A	K	iter	δ_1	δ_2
0.2	0.1	0.02	100	10	0.0006	10^{-28}
0.2	0.1	0.02	200	8	0.0031	10^{-20}
0.2	0.1	0.02	400	9	0.0131	10^{-18}

Table 3: Approximation of one sinusoidal component.

ω	φ	A	K	iter	δ_1	δ_2
0.1 0.2	−0.1 0.1	3 2	100	18	0.5452	10^{-17}
0.1 0.2	−0.1 0.1	3 2	200	18	0.0194	10^{-19}
0.1 0.2	−0.1 0.1	3 2	400	20	0.0727	10^{-16}

Table 4: Approximation of two sinusoidal components.

ω	φ	A	K	iter	δ_1	δ_2
0.85 0.98	−0.1 0.1	1201 1200	100	19	7.6271	10^{-10}
0.85 0.98	−0.1 0.1	1201 1200	200	17	19.470	10^{-13}
0.85 0.98	−0.1 0.1	1201 1200	400	20	77.481	10^{-11}

Table 5: Approximation of two sinusoidal components.

ω	φ	A	K	iter	δ_1	δ_2
0.1 0.2	−0.1 0.1	1800 179	100	20	568.43	10^{-12}
0.1 0.2	−0.1 0.1	1800 179	200	20	14.969	10^{-10}
0.1 0.2	−0.1 0.1	1800 179	400	20	118.35	10^{-12}

Table 6: Approximation of two sinusoidal components.

ω	φ	A	K	iter	δ_1	δ_2
1 0.1 0.2	0 −0.1 0.1	30 3 2	100	27	0.7118	10^{-16}
1 0.1 0.2	0 −0.1 0.1	30 3 2	200	21	0.0723	10^{-13}
1 0.1 0.2	0 −0.1 0.1	30 3 2	400	22	0.0706	10^{-15}

Table 7: Approximation of three sinusoidal components.

ω	φ	A	K	iter	δ_1	δ_2
1 1.2 1.1	0 0.5 −0.9	1200 200 129	100	28	2178.4	10^{-12}
1 1.2 1.1	0 0.5 −0.9	1200 200 129	200	21	41.797	10^{-12}
1 1.2 1.1	0 0.5 −0.9	1200 200 129	400	25	57.899	10^{-10}

Table 8: Approximation of three sinusoidal components.

ω	φ	A	K	iter	δ_1	δ_2
1	0	130				
0.1	−0.1	129	100	29	0.3769	10^{-12}
0.2	0.1	128				
1	0	130				
0.1	−0.1	129	200	25	0.3253	10^{-13}
0.2	0.1	128				
1	0	130				
0.1	−0.1	129	400	26	0.4165	10^{-12}
0.2	0.1	128				

Table 9: Approximation of three sinusoidal components.

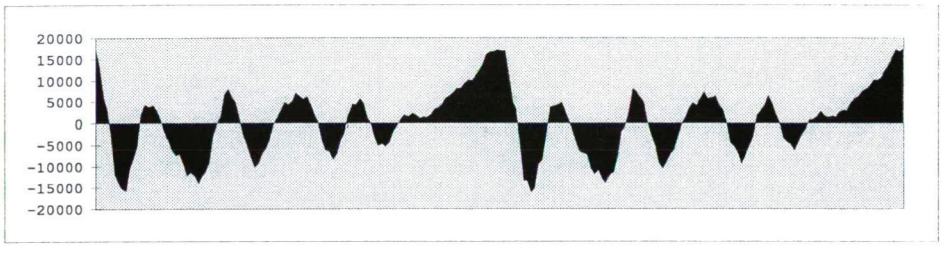


Figure 1:

provided the sinusoidal decomposition of the signal and the sound pattern was synthesised from the obtained sinusoidal components. The parameters of the first 20 dominant sinusoidal components of the DFT decomposition were used as initial parameters of the NHSRF optimization scheme and the minimization of (6) yielded the optimum decomposition of /*a*/, by the NHSRF-based procedure. The sound signal was synthesised again from the obtained components. Unfortunately the quality of sound synthesis is not easy to measure, since the metric is not Euclidean, but a 'perceptual' distance function would be necessary to measure the 'goodness' of the representation procedure. Therefore the 'comparison by listening' of the original and synthesised sounds had a decisive role in the judgement. However to give an easily noticeable impression on the accuracies of approximations of the natural sound pattern, figure 1-3 display the original signal, the signal synthesised from the 50 largest amplitude components of the DFT-based decomposition and the signal synthesised from the 20 components of the NHSRF-based decomposition.

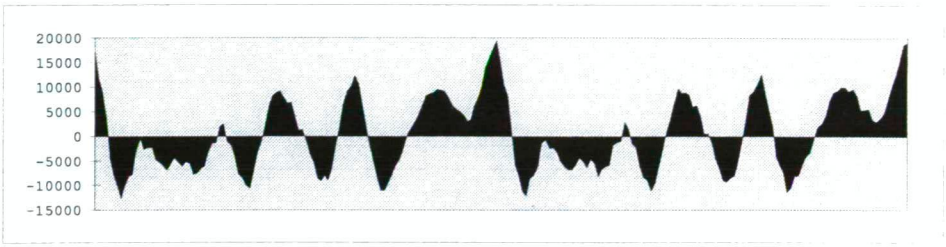


Figure 2:

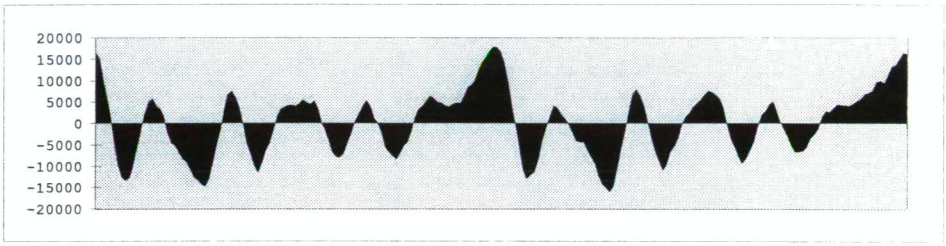


Figure 3:

7 Discussion

7.1 Artificial signals

The application of the proposed algorithm is especially important if either the components of the signal are required with high accuracies, or the usual DFT-based technics are not suitable to provide acceptably accurate results at all. This case occurs if the sample is too short, the components are too close, or the amplitudes differ too much. Our test functions were therefore of these types.

The proposed algorithm proved to be powerful in correcting the estimations of the DFT-based decomposition procedure even in those cases where the former algorithm provided quite acceptable results. Notice that the DFT-based scheme estimated the phases very poorly, while the NHSRF-based algorithm corrected these values.

7.2 Real sound signals

The proposed algorithm allowed to reconstruct the analysed sound signal from less sinusoidal components (20 components) at higher accuracy than the DFT-based method (50 components), as seen clearly on the figures. The DFT-based decomposition of voiced sounds is generally unable to provide the sinusoidal fundamental and overtones accurately enough and yields more overtone components than the sample really contains. However the high-fidelity modelling of voiced sounds re-

quires to synthesise the signal by the least sinusoidal components defined with accurately determined parameters. This is a necessary condition for developing and using efficient data compression technics, too.

7.3 Future Work

On the basis of the presented results, the NHSRF-optimizing algorithm proved to be robust and efficient in applications of speech- and audio-processing. The aim is to use the algorithm in sound coding and to develop a more advanced pitch estimation method than the ones used nowadays. The momentarily fixed dimension of the approximation subspace will be handled as variational parameter in the future. This feature will help to separate the sinusoidal and the noiselike components of the sound allowing to screen noise, to detect the unvoiced/voiced parts of the sound, furthermore the upgraded procedure would be a candidate for being applied in those sound coding methods, which are based on the 'sinusoidal + noise'-type decomposition of the signal (e.g. Quatieri-McAulay).

References

[1] Alan V. Oppenheim, Ronald W. Schafer: Discrete-Time Signal Processing, PRENTICE HALL

[2] L.R. Rabiner, R.W Schafer: Digital Processing of Speech Signals, PRENTICE HALL

[3] S. Lawrence MArple, Jr. : Digital Spectral Analysis with applications, PRENTICE HALL

[4] McAulay, R.J. and T.F. Quatieri. 1986. "Speech Analysis/Synthesis based on a Sinusoidal Representation." IEEE Transactions on Acoustics, Speech and Signal Processing, 34(4):744–754

[5] Allen, J.B 1977. "Short Term Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transformation. " IEEE Transactions on Acoustics, Speech and Signal Processing, 25(3):235-238

[6] Hess, W. 1983. Pitch Determination of Speech Signals. New york: Springer-Verlang

[7] Harrs, F.J. 1978. " On the use of windows for harmonic analysis with the discrete Fourier transform." Proceedings IEEE, vol. 66, pp.51-83.

[8] Goodwin, M. and X.Rodet. 1994. "Efficient Fourier Synthesis of Nonstacionary Sinusoids." Proceedings of the 1994 International Computer Music Conference. San Francisco: Computer Music Association.

[9] Mather, R. C. and J. W. Beauchamp. 1994. "Fundamental Frequency Estimation of Musical Signals using a two-way Mismatch Procedure." Journal of Acoustical Society of America 95(4):2254–2263

[10] Mokhtar S. Bazaraa, Hanif D. Sherali, C. M. Shetty, *NONLINEAR PROGRAMMING theory and algorithms*, John Wiley & Sons, [1993]

[11] J. L. Nazareth, *Conjugate Gradient Methods Less Dependency on Conjugacy*, SIAM Review, 28(4), PP. 501-511, 1986.

[12] J. Nocedal, *The Performance of Several Algorithms for Large Scale Unconstrained Optimization, in Large-Scale Numerical Optimization*, T. F. Coleman and Y. Li (Eds.), SIAM, Philadelphia, pp. 138-151, 1990.

[13] Gill, P.E., W. Murray, and P.A. Pitfield, *The implementation of Two Revised Quasi-Newton Algorithms for Unconstrained Optimization*, Report NAC-11, National Physical Lab., 1972

[14] Usmani, R. A.: Applied Linear Algebra. Marcel Dekker, New York, 1987

[15] Lippmann, Stanley B.: C++ Primer. Addison Wesley, 1991

[16] A. Kocsor, J. Dombi, I. Bálint, An Optimization Algorithm for Determining Eigenpairs of Large Real Matrices, (submitted to SIAM Journal On Scientific Computing)

Acta Cybernetica14 (1999) 331-339.

Improved Greedy Algorithm for Computing Approximate Median Strings

Ferenc Kruzslicz *

Abstract

The distance of a string from a set of strings is defined by the sum of distances to the strings of the given set. A string that is closest to the set is called the median of the set. To find a median string is an NP-Hard problem in general, so it is useful to develop fast heuristic algorithms that give a good approximation of the median string. These methods significally depend on the type of distance used to measure the dissimilarity between strings. The present algorithm is based on edit distance of strings, and constructing the approximate median in a letter by letter manner.

1 Introduction

If the solution of the optical character recognition (OCR) problem is considered as a "black box" process where images are mapped to character strings, then we usually use a certain kind of off-line approach. In this way the efficiency of some OCR processes could be increased in an OCR software and language independent manner. Suppose we have a set of strings as the result of several OCR processes of the same input bitmap. When the same OCR software was used to produce this set, with different paper orientation, changed resolution or simply repeated OCR processes we can eliminate the effects of noise (fingerprints on the glass etc.). While in case of different OCR software their efficiency can be compared to each other [7].

2 String distance

Finding a median string that is minimal in sum of distances form a given input set of strings, is known to be an NP-hard problem [8]. Therefore it is interesting to find fast algorithms, that give us good approximations. One of the latest algorithms can be found in [3]. It is called greedy algorithm, because it builds up the approximate median string letter by letter, by always choosing the best possible continuation. In this paper an improvement of this algorithm is described.

*Department of Business Informatics, Janus Pannonius University, Rákóczi út 80, 7622 Pécs, Hungary. Email: kruzslic@ktk.jpte.hu

Suppose that all the strings are defined over the same fixed alphabet Σ (for European countries Σ is usually a certain kind of extended ASCII). The most widely used edit distances are similar to the Levenshtein distance. The improved greedy algorithm is based on the dynamic programming approach [4], therefore it is suitable for all $d : (\Sigma^*)^2 \to R$ distances that satisfies the following properties.

$d(t,s) \geq 0$
$d(t,s) = 0 \Longleftrightarrow t = s$
$d(s,t) = d(t,s)$
$d(s,r) + d(r,t) \geq d(s,t)$
for all $r, s, t \in \Sigma^*$.

In case of $c \in \Sigma$ let $c_{ins}(c,r)$, $c_{del}(c,r)$, $c_{sub}(c,r)$ denote the cost of insertion, deletion and substitution of letter c in string r. The costs of edit operations do not depend on letter c and on the place of operations in r.

The Levenshtein distance is derived from this class of distances by choosing the following values: $c_{ins}(c,r) = c_{del}(c,r) = c_{sub}(c,r) = 1$. To establish the Levenshtein distance between two strings, the dynamic programming approach can be used with $O(nm)$ time and $O(n)$ space complexity. The general algorithm to compute the minimal edit distance, using the dynamic programming technique is given in the paper of Kruskal [5]. With the aid of this method, we get the following in the case of two strings (s and t):

```
Let D[i,0] = i and D[0,j] = j for i=0..|s| and j=0..|t|.
For i=1..|s| and j=1..|t| calculate the next elements of matrix D
   D[i,j] = min (D[i-1,j]+ c_ins, D[i,j-1]+ c_del, D[i-1,j-1]+δ([i,j]), where
        δ([i,j] = c_sub if s[i]≠t[j], and 0 otherwise.
```

It is clear that the distance is $d(s,t) = D[|s|,|t|]$.
Much space can be saved if the matrix D is computed in a row by row manner.

3 Approximate median

This dynamic programming technique is suitable for a large number of heuristics. Almost all of the "natural" heuristics can be described by the following informal scheme, where $|r|$ denotes the length of r, and λ is the empty string.

```
function ApproximateMedian(s1,s2 ... sn): string;
   preprocess(s1, s2, ... sn);
   median = λ;
   do
      c_best = arg best (weight(median, c, s1, s2, ..., sn) : c ∈ Σ);
      median = median + c_best;
   while ( it was worth to append c_best );
return( best prefix of median ).
```

Basically, an ApproximateMedian algorithm of this type builds the median string letter by letter, and in case of each letter it uses a weight decision function to select the next letter for median string to continue with. It makes judgements on the base of input strings $s_1, s_2, ..., s_n$ and the prebuilt median appended with letter c. The previous loop has to be continued, until a stopping condition holds. In the last step we can select the best prefix of median to return.

The time and space complexity of these algorithms is determined by the complexity of the preprocessing phase and the weight function. The previous scheme is general enough, because any type of algorithm can be written in this high level form. In case of greedy algorithms no preprocessing phase is allowed, and the weight function must be linear.

4 The Improved Greedy Algorithm

The earlier scheme of algorithms gives a large variety of heuristics. We have freedom to choose the weight functions, the stopping condition, and the last prefix correction.

A fairly good greedy heuristic can be obtained if we use the method in [3], i.e.

- The weight function is the sum of minimal elements in the last rows containing letter c in the dynamic programming matrix, computing $d(median + c, s_i)$. The next letter to be appended is the letter with the minimal weight.
- The main loop is stopped if the length of median reaches the length of the longest input string.
- The prefix of median is returned, that minimise the sum of distances from the input strings.

The greedy algorithm computes the whole dynamic programming matrixes, but stores only the last rows of them, and it loses a lot of information, because it uses only the minimal element of this vector. Let the algorithm improve by gaining more information from this vector.

If we sum these vectors, we get information on what would happen if we stop the algorithm immediately. The values of the summed vector show the sum of distances of median from the input strings, and the sum distances of median from the input strings without their last letter, etc. For example strings *aabb*, *ab*, *bbb* and median string *ab* will be examined:

b	2	1	1	1	2
a	1	0	1	2	3
λ	0	1	2	3	4
	λ	a	a	b	b

b	2	1	0	
a	1	0	1	
λ	0	1	2	
	λ	a	b	

b	2	1	1	2	
a	1	1	2	3	
λ	0	1	2	3	
	λ	b	b	b	

The sum of the last rows of matrixes D is defined as follows:

aabb	2	1	1	1	2
ab			2	1	0
bbb		2	1	1	2
S	2	3	4	3	4

or more precisely, let m denote the length of median string, and $k = max(|s_1|, |s_2|, ..., |s_n|)$. Moreover the last row of the ith matrix is denoted by $V_i =< D[|m|, 0], D[|m|, 1], ..., D[|m|, s_i|] >$. For convenience, we also assume that the co-ordinate $V_i[t] = 0$, whenever t is not in the $0...|s_i|$ interval. The summarised vector S is defined with the following expression

$$S[i] = \sum_{j=1}^{n} V_i[i - k + |s_i|], \text{ for } i = 0, ..., k.$$

With these notations the weight function in the greedy algorithm can be formulated in a simple way:

$$weight(median, c, s_1, s_2, ..., s_n) = \sum_{j=1}^{n} min(V_j[0], V_j[1], ..., V_j[|s_j|])$$

and the letter with the least weight will be appended to the median string.

Unfortunately this weight function frequently gives the same value for different letters, and in such a case the next letter is selected arbitrary. The weight function behaves better if we use the whole V vector to pick the best continuation of the median. Let us choose the letter in case of draw, that is minimal in lexicographic order of the reversed sum vectors $< S[k]; S[k-1]; ...; S[0] >$. Clearly the choice of next letter tries to minimise the expected sum of distances, furthermore the time and space complexity of the algorithm remains the same.

The improved algorithm runs in $O(k^2 n|\Sigma|)$ time, and it is given in the following pseudocode.

```
function ImprovedApproximateMedian(s1, s2, ..., sn) : string;
   constants
      k = max( |s1|, |s2|, ..., |sn|);
      c_ins, c_del, c_sub;     /* Cost of edit operations */
   variables
      Vi : array [0..|si|] of integer; /* for i=1..n */
      Dist, S, S_best, tmp : array [0..k] of integer;
      c : char;
      min_best, min_sum, i, j : integer;
      median : array [1..k] of char;
   algorithm
      median = λ;              /* Initialization */
      Dist[0] = 0;
      for i=1 to n do
```

```
         for j=0 to |s_i| do V_i[j] = j; od
         Dist[0] = Dist[0] + |s_i|;
      od
      for i=1 to k do          /* Building the median letter by letter */
         S_best := [0,0,...,0];
         for j=1 to n do S_best := add_vect(S_best, V_j, k - |s_j|); od
         for each c ∈ Σ do       /* Selecting the best letter */
            min_sum := 0;
            S := [0,0,...,0];
            for j=1 to n do test_letter(c, j, FALSE); od
            min_sum := min_sum + min_best;
            S := add_vect( S, tmp, k - |s_i| )
            if weight(min_sum, S, min_best, S_best) < 0 then
               S_best := S;
               min_best := min_sum;
               median[i] := c;
               Dist[i] := S_best[k];
            fi
         od
         for j=1 to n do test_letter(median[i], j, TRUE); od
      od
      i := 0;
      for j=1 to k do
         if Dist[j] < Dist[i] then i = j; fi
      od
return median[1..i]

function test_letter(c, i, update) : integer;
   local variables
      j : integer;
   procedure                  /* Calculating the edit distance */
      min_best := +∞;
      tmp[0] := i;
      for j=1 to |s_i| do
         tmp[j] := min( V_i[j-1]+c_ins, V_{i-1}[j]+c_del, V_{i-1}[j-1]+c_sub );
         if median[j] = c then
            tmp[j] = min( tmp[j], V_{i-1}[j-1] );
         fi
         if tmp[j] > min_best then
            min_best = tmp[j];
         fi
      od
      if update then          /* Updating vectors when a */
         V_i := tmp[0..|s_i|];    /* new letter was appended. */
      fi
return min_best;

function add_vect(S,V,offset) : array [0..k] of integer;
   local variables
      i : integer
   procedure        /* Vector addition with offset */
      for i=0 to k - offset do
         S[i] := S[i] + V[i-offset];
return S[0..k];
```

```
function weight(min, S, min_best, S_best) : boolean;
   local variables
      diff, i : integer    /* Negative value is returned if the new */
   procedure               /* character is better than the old one. */
      diff := min - min_best;              /* Greedy heuristic */
      i := k;
      while ( i ≥ 0 and diff = 0 ) do      /* Lexicographic order */
         diff := S[i] - S_best[i];
         i := i-1;
       od
return diff
```

To illustrate how the algorithm works and to show the improvement, let us examine the following example:

The alphabet contains only two letters $\Sigma = \{a, b\}$, and the input strings are $s_1 = ab$, $s_2 = bab$.

a	1	**0**	1
λ	0	1	2
	λ	a	b

a	1	**1**	1	2
λ	0	1	2	3
	λ	b	a	b

b	1	**1**	1
λ	0	1	2
	λ	a	b

b	1	**0**	1	2
λ	0	1	2	3
	λ	b	a	b

It is easy to see that we are in the draw situation, since for

median = a, min_sum = 1, S = ¡1,2,1,3¿, and for

median = b, min_sum = 1, S = ¡1,1,2,3¿.

By the rule of the improved greedy algorithm letter a will be selected as the first letter of the median string.

5 Experimental Results

The improved approximate algorithm was tested on the same garbled strings as the greedy algorithm. In the test sets the string were deformed with equally probable delete, insert and substitute operations, with probability of 1/4.

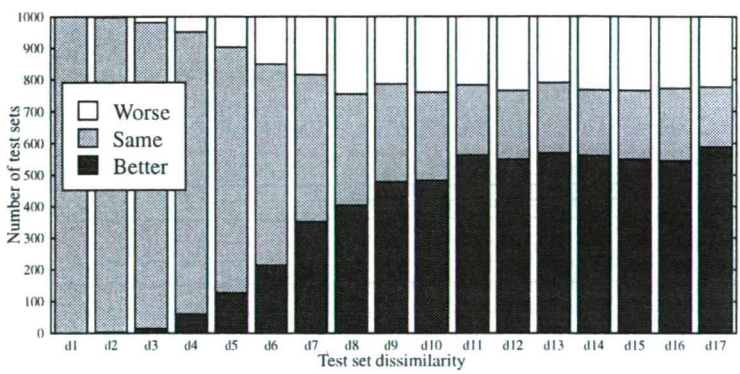


Figure 1: Efficiency of the improved versus the greedy algorithm

Original words:

hector	helsinki	iapr	ojo	pepermint	recognition	sim patica

Garbled strings:

erth	eksh	arr	jj	etepi	etorgon	icpsa
ttohr	ielkhnnki	rpp	oo	petpnrmin	tricogntionr	sic static
ectoo	hlsinhki	iaria	j	pepeprmiimtn	recggginiiong	simpsatiapat
heceor	hislnsiki	iaprr	ojjo	peermmint	receniicion	sipatpica
htoor	hselsekni	iapri	jjo	epneemine	egcoogeieion	imtpitici
ecttor	eelseskli	iappp	oj	merpeement	regtoggniitocn	pimmpitaca
hetroe	hlliinki	irap	ojo	pepitrmminnt	recortoit	siaatpta
hecetrc	hiklssinnksl	iappr	oojo	mrpermimm	ecgnittin	satica
heeter	elsinss	iai	oooj	eentin	reoritoc	pppttca
hectter	esnkki	iraar	oj	pepterintm	enoeniiion	smpactia
hector	helsinki	iapr	ojo	pepermint	recognition	simptatica

Greedy approximate medians:

hector	helsinki	iapr	ojo	pepermint	recognition	simp tatica
[26]	[39]	[19]	[12]	[39]	[50]	[45]

Improved approximate medians:

hector	helsinki	iapr	ojo	pepermint	recogniion	simptatica
[26]	[39]	[19]	[12]	[39]	[47]	[45]

When we used the new algorithm for the second test sets published in [6], there were no improvements at all.

Original words:

hector	helsinki	iapr	recognition

Garbled strings:

hetcr	cheinni	cianr	rgfkfgnition
heptor	hlelsiki	iap	recoxsniimoi
hector	hesenkc	iapi	riecoxgnifon
hevor	velskki	lapr	jeognitigqn
hetuor	ceeltsinkmi	ilp	resonigior
hscor	elnsgxnki	riapr	reoinitiggn
htuctor	gbheklsink	ialr	rciorgnitvihn
fjhecto	htosini	iar	recognin
getoqr	hxlsiky	iapd	ecotnritiin
hetofr	heklusnkk	iuar	grecpoginitko

Greedy and improved approximate medians:

hector	helsinki	iapr	recognition
[13]	[34]	[13]	[42]

The real advantage of the improved algorithm appeared when the probability of the edit operations has been increased. The Fihure 1 is obtained by the following test sets. The string *recognition* was garbled with delete, insert and substitute edit operations. For substituting and inserting only the letters *r, e, c, g, t, i, o, n, s, p, a* were used, and each of the operations and its place was uniformly distributed.

Every test consists of 10 garbled strings, and the index of a test means how many operations was performed in the garbling procedure. All column of the diagrams represents the results of 1000 tests of the same type. The greedy and the improved algorithms were compared, the bars show in how many cases which one was the better. In some cases the greedy algorithm proved to be superior to the improved one. The reason for this is, that in a case of draw in the greedy algorithm the next letter was chosen randomly, that could result in a better performance.

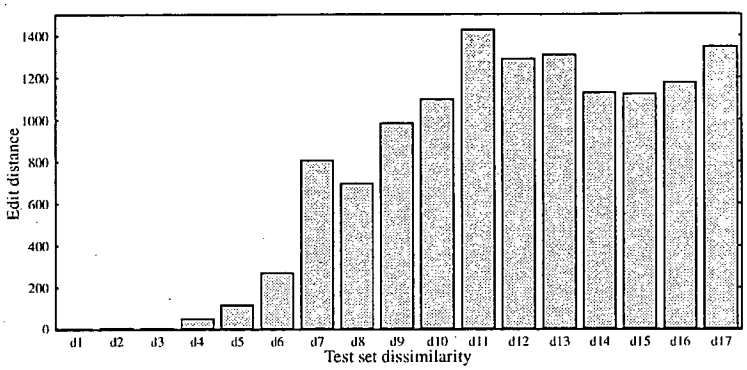


Figure 2: Improvements measured in edit distance.

In Figure 2 the total distances were summed (i.e. the distances of the approximate median from the test set). The same garbled sets were used as in Figure 1 and values of diagram are the difference between the totals for the improved and the greedy algorithm. We see that a slight modification in the greedy algorithm results in computing better medians whenever the problem becomes more difficult. Since the total sum of distances is bounded from above by the total length of strings from the test set, the results remain stable when we choose the dissimilarity value higher than the length of the distorted string.

6 Conclusions

The improved approximate median algorithm is a simple refinement of the greedy algorithm [3]. It has the same time complexity $O(k^2 n|\Sigma|)$ as the previous one. The space complexity was a bit reduced by the help of storing only the last rows of the distance matrixes. This idea is based on [9], in this way the new algorithm runs in $O(kn)$ space. The closer the garbled strings are to each other the improvement is less significant. Therefore the improved algorithm presented in this paper is more suitable for searching approximate median of highly dissimilar strings.

Acknowledgement

I am grateful to the anonymous reviewers for their helpful comments which helped me improve the quality of the paper. In particular, I thank the anonymous referee who provided an improved English version of my manuscript, and Dr. János Csirik for calling my attention to the median string problem.

References

[1] D. Lopestri, J. Zhou: Using Consensus Voting to Correct OCR Errors. Series in Machine Perception and Artifical Intelligence Vol. 14 pages 157-168, 1995

[2] A. Juan, E. Vidal: An Algorithm For Fast Median Search. Pattern Recognition and Image Analysis Vol. 1 pages 187-192, 1996

[3] F. Casacuberta, M. D. Antonio: A greedy algorithm for computing approximate Median Strings. Pattern Recognition and Image Analysis Vol. 1 pages 193-198, 1996

[4] M. Crochemore, W. Rytter: Text Algorithms. Oxford University Press, 1994

[5] J.B. Krushkal: An overview of sequence comparison: Time warps, string edits, and macromolecules. SIAM Review Vol. 25 pages 201-237, 1983.

[6] H. Rulot: Un Algoritmo de Inferencia Gramatical mediante Correccin de Errores. Tesis Doctoral. Universitat de Valencia, 1992.

[7] S. V. Rice, J. Kanai, T. A. Nartker: A difference algorithm for OCR-generated text. Proceeding of the IAPR Workshop on Structural and Syntactic Pattern Recognition, Bern, 1992.

[8] T. Kohonen: "Median strings". Pattern Recognition Letters Vol. 3 pages 309-313, 1985.

[9] L. Allison, T. I. Dix: A bit-string longest-common-subsequence algorithm. Information Processing Letters Vol. 23 pages 305-310, 1986

Acta Cybernetica **14** (1999) 341-356.

Limiting Distortion of a Wavelet Image Codec

Joonas Lehtinen *

Abstract

A new image compression algorithm, *Distortion Limited Wavelet Image Codec* (DLWIC), *is introduced.* The codec is designed to be *simple to implement, fast* and have *modest requirements for the working storage.* It is shown, *how the distortion of the result can be calculated while progressively coding a transformed image* and thus how the *mean square error of the result can be limited* to a predefined value. The DLWIC uses zerotrees for efficient coding of the wavelet coefficients. Correlations between different orientation components are also taken into account by binding together the coefficients on the three different orientation components in the same spatial location. The maximum numbers of significant bits in the coefficients of all subtrees are stored in two-dimensional heap structure that allows the coder to test the zerotree property of a subtree with only one comparison. The compression performance of the DLWIC is compared to the industry standard JPEG compression and to an advanced wavelet image compression algorithm, vqSPIHT. An estimation of execution speed and memory requirements for the algorithm is given. The compression performance of the algorithm seems to exceed the performance of the JPEG and to be comparable with the vqSPIHT.

1 Introduction

In some digital image archiving and transferring applications, especially in medical imaging, the quality of images must meet predefined constrains. The quality must be often guaranteed by using a lossless image compression technique. This is somewhat problematic, because the compression performance of the best known lossless image compression algorithms is fairly modest; the compression ratio ranges typically from 1:2 to 1:4 for medical images [5].

Lossy compression techniques generally offer much higher compression ratios than lossless ones, but this is achieved by losing details and thus decreasing the quality of the reconstructed image. Compression performance and also the amount of distortion are usually controlled with some parameters which are not directly connected to image quality, defined by *mean square error* [1] (MSE). If a lossy technique is used, the quality constrains can be often met by overestimating the control parameters, which results worse compression performance.

*Turku Centre for Computer Science, University of Turku, Lemminkäisenkatu 14 A, 20520 Turku, Finland, email: jole@jole.fi, WWW: http://jole.fi/

In this paper a new lossy image compression technique called *Distortion Limited Wavelet Image Codec* (*DLWIC*) is presented. The DLWIC is related to *embedded zerotree wavelet coding* (EZW) [9] technique introduced by J.M. Shapiro in 1993. Also some ideas from SPIHT [8] and vqSPIHT [3] have been used. DLWIC solves the problem of *distortion limiting* (DL) by allowing the user of the algorithm to specify the *MSE of the decompressed image as controlling parameter for the compression algorithm.*

The algorithm is designed to be *as simple as possible*, which is achieved by *binding together the orientation bands of the octave band composition* and *coding the zerotree structures and wavelet coefficient bits in the same pass.* A special auxiliary data structure called *two dimensional heap is introduced* to make the zerotree coding simple and fast. The DLWIC uses only little extra memory in the compression and is thus suitable for compression of very large images. The technique also seems to provide competitive compression performance in comparison with the vqSPIHT.

In the DLWIC, the image to be compressed is first converted to the wavelet domain with the orthonormal *Daubechies wavelet transform* [10]. The transformed data is then coded by bit-levels using a scanning algorithm presented in this paper. The output of the scanning algorithm is coded using *QM-coder* [7], an advanced binary arithmetic coder.

The scanning algorithm processes the bits of the wavelet transformed image data in decreasing order of their significance in terms of MSE, as in the EZW. This produces *progressive* output stream: the algorithm can be stopped at any phase of the coding and the already coded output can be used to construct an approximation of the original image. This feature can be used when a user browses images using slow connection to the image archive: The image can be viewed immediately after only few bits have been received; the subsequent bits then make it more accurate. The DLWIC uses the progressivity by stopping the coding when the quality of the reconstruction exceeds threshold given as a parameter to the algorithm. The coding can also be stopped when the size of the coded output exceeds a given threshold. This way both the MSE and *bits per pixel* (BPP) value of the output can be accurately controlled.

After the introduction, the structure of the DLWIC is explained. A quick overview of the octave band composition is given and it is shown with an example how the wavelet coefficients are connected to each other in different parts of the coefficient matrix.

Some general ideas of the bit-level coding are then explained (2.3) and it is shown how the unknown bits should be approximated in the decoder. The meaning of zerotrees in DLWIC is then discussed (2.4). After that an auxiliary data structure called two dimensional heap is introduced (2.5). The scanning algorithm is given as pseudo code (2.6).

The distortion limiting feature is introduced and the stopping of the algorithm on certain stopping conditions is discussed (2.7). Finally we show how separate probability distributions are allocated for coding the bits with the QM-coder in different contexts (2.8).

The algorithm is tested with a set of images and the compression performance

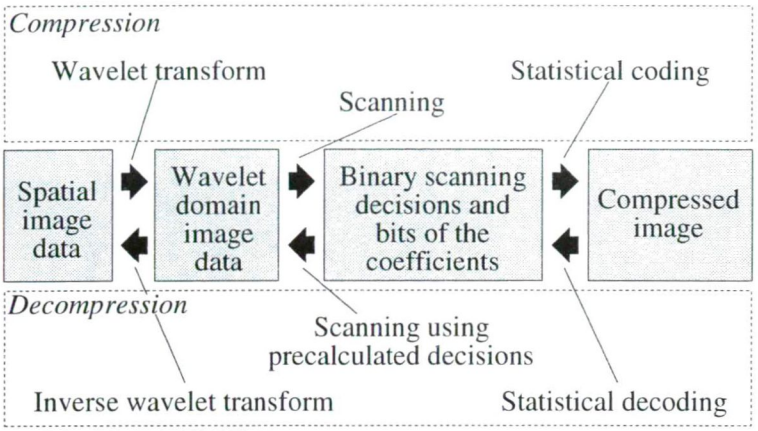


Figure 1: The structure of the DLWIC compression algorithm

is compared to the JPEG and the vqSPIHT compression algorithms (3). Variations in the quality achieved by the constant quantization in the JPEG is demonstrated with an example. Also an estimation of the speed and memory usage is given (3.2).

2 DLWIC algorithm

2.1 Structure of the DLWIC and the wavelet transform

The DLWIC algorithm consists of three steps (Figure 1): 1) the wavelet transform, 2) scanning the wavelet coefficients by bit-levels and 3) coding the binary decisions made by the scanning algorithm and the bits of the coefficients with the statistical coder. The decoding algorithm is almost identical: 1) binary decisions and coefficient bits are decoded, 2) the coefficient data is generated using the same scanning algorithm as in the coding phase, but using the previously coded decision information, 3) the coefficient matrix is converted to a spatial image with the inverse wavelet transform.

The original spatial domain picture is transformed to the wavelet domain using Daubechies wavelet transform [10]. The transform is applied recursively to the rows and columns of the matrix representing the original spatial domain image. This operation gives us an octave band composition (Figure 2). The left side (B) of the resulting coefficient matrix contains horizontal components of the spatial domain image, the vertical components of the image are on the top (A) and the diagonal components are along the diagonal axis (C). Each *orientation pyramid* is divided to levels, for example the horizontal orientation pyramid (B) consists of three levels (B0, B1 and B2). Each level contains details of different size; the lowest level (B0), for example, contains the smallest horizontal details of the spatial image. The three orientation pyramids have one shared top level (S), which contains

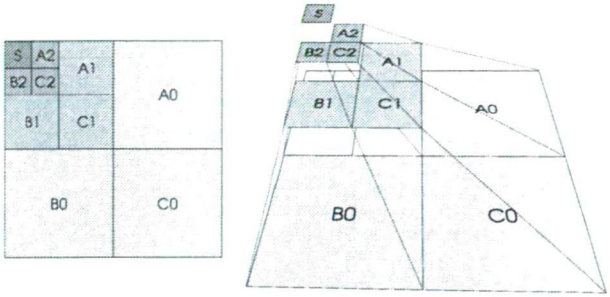


Figure 2: Octave band composition produced by recursive wavelet transform is illustrated on the left and the pyramid structure inside the coefficient matrix is shown on the right.

scaling coefficients of the image, representing essentially the average intensity of the corresponding region in the image. Usually the coefficients in the wavelet transform of a natural image are small on the lower levels and bigger on the upper levels (Figure 3). This property is very important for the compression: the coefficients of this highly skewed distribution can be coded using fewer bits.

2.2 Connection between orientation pyramids

Each level in the coefficient matrix represents certain property of the spatial domain image in its different locations. Structures in the natural image contain almost always both big and small details. In the coefficient matrix this means that if some coefficient is small, it is most likely that also the coefficients, representing smaller features of the same spatial location, are small. This can be seen in Figure 3: different levels of the same coefficient pyramid look similar, but are in different scales. The EZW takes advantage of this by scanning the image in depth first order, i.e. it scans all the coefficients related to one spatial location in one orientation pyramid before moving to another location. This way it can code a group of small coefficients together, and thus achieves better compression performance.

In natural image, most of the features are not strictly horizontal or vertical, but contain both components. The *DLWIC takes advantage of this by binding also all three orientation pyramids together*: The scanning is done only for the horizontal orientation pyramid (B), but bits of all three coefficients, representing the three orientations of the same location and scale, are coded together. Surprisingly this only slightly enhances the compression performance. The feature is however included in the DLWIC because of its advantages: it simplifies the scanning, makes the implementation faster and reduces the size of auxiliary data structures.

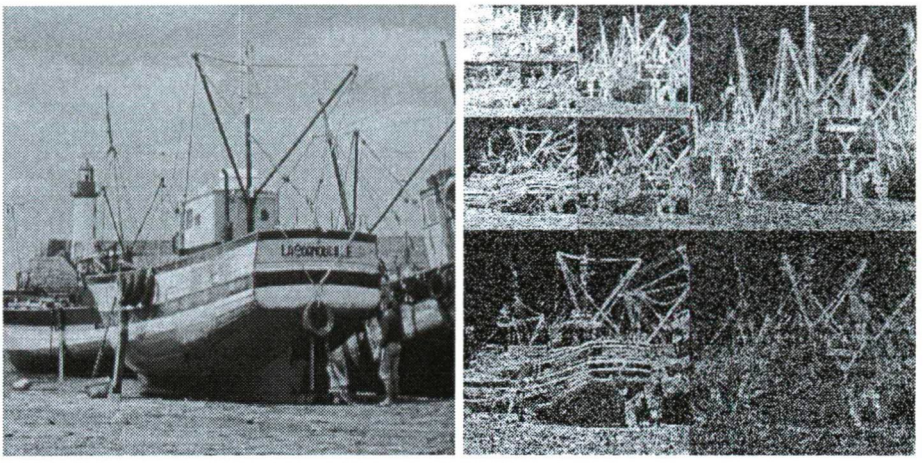


Figure 3: An example of the Daubechies wavelet transform. The original 512×512 sized picture is on the left and its transform is presented with absolute values of the coefficients in logarithmic scale on the right.

2.3 Bit-level coding

The coefficient matrix of size $W \times H$ is scanned by bit-levels beginning from the highest bit-level n_{max} required for coding the biggest coefficient in the matrix (i.e. the number of the significant bits in the biggest coefficient):

$$n_{max} = \lfloor \log_2(\max\{(|c_{i,j}|) | 0 \leq i < W \wedge 0 \leq j < H\}) + 1 \rfloor, \tag{1}$$

where the coefficient in (i, j) is marked with $c_{i,j}$. The coefficients are represented using positive integers and the sign bits that are stored separately. The coder first codes all the bits on the bit-level n_{max} of all the coefficients, then all the bits on bit-level $n_{max} - 1$ and so on until the least significant bit-level 1 is reached or the scanning algorithm is stopped (Section 2.7). The sign is coded together with the most significant bit (the first 1-bit) of a coefficient. For example three coefficients $c_{0,0} = -19_{10} = -10011_2, c_{1,0} = 9_{10} = 01001_2, c_{2,0} = -2_{10} = -00010_2$ would be coded as

$$\underbrace{1100}_{5}\underbrace{0100}_{4}\underbrace{000}_{3}\underbrace{1011}_{2}\underbrace{110}_{1}, \tag{2}$$

where the corresponding bit-level numbers are marked under the bits coded on that level (without signs it would be $\underbrace{100}_{5}\underbrace{010}_{4}\underbrace{000}_{3}\underbrace{101}_{2}\underbrace{110}_{1}$).

Because of the progressivity, the code stream can be truncated at any position and the decoder can approximate the coefficient matrix using received information. The easiest way of approximating the unknown bits in the coefficient matrix would be to fill them with zeroes. In the DLWIC algorithm a more accurate estimation is

used, the first unknown bit of each coefficient, for which the sign is known, is filled with one and the rest bits are filled with zeroes. For example, if the first seven bits of the bit-stream (2) have been received, the coefficients would be approximated: $c_{0,0} = -20_{10} = -10100_2, c_{1,0} = 12_{10} = 01100_2, c_{2,0} = 0_{10} = 00000_2$.

2.4 Zerotrees in DLWIC

A bit-level is scanned by first coding a bit of a scaling coefficient (on the level S in the Figure 2). Then recursively the three bits of the coefficients in the same spatial location on the next level of the orientation pyramids (A2,B2,C2) are coded. The scanning continues to the next scaling coefficient, after all the coefficients in the previous spatial location in all the pyramid levels has been scanned.

We will define that a coefficient c is *insignificant* on a bit-level n, if and only if $|c| < 2^{n-1}$. Because the coefficients on the lower pyramid levels tend to be smaller than on the higher levels and different sized details are often spatially clustered, probability for a coefficient for being insignificant is high, if the coefficient on the higher level in the same spatial location is insignificant.

If an insignificant coefficient is found in the scanning, the compression algorithm will check if any of the coefficients below the insignificant one is significant. If no significant coefficients are found, all the bits in those coefficients on current bit-level are zeroes and thus can be coded with only one bit. This structure is called *zerotree.*

One difference to the EZW algorithm is that the DLWIC scans all the orientations simultaneously and thus constructs only one shared zerotree for the all the orientation pyramids. Also the significance information is coded at the same pass as the significant bits in the coefficients, whereas the EZW and SPIHT algorithms use separate passes for the significance information.

2.5 Two dimensional significance heap

It is a slow operation to perform a significance check for all the coefficients on a specific spatial location on all the pyramid levels. The DLWIC algorithm uses a new auxiliary data-structure, which we call *two dimensional significance heap*, to eliminate the slow significance checks.

The heap is a two dimensional data-structure of the same size (number of elements) and shape as the horizontal orientation pyramid in the coefficient matrix. *Each element in the heap defines the number of bits needed to represent the largest coefficient in any orientation pyramid in the same location on the same level or below it.* Thus the scanning algorithm can find out, whether there is a zerotree starting from a particular coefficient on a certain bit-level by comparing the number of the bit-level to the corresponding value in the heap.

Here and in the rest of this paper we denote the height of the coefficient matrix with H, the width with W and the number of levels in the pyramid excluding the scaling coefficient level (S) with L. Thus the dimensions of the scaling coefficient level are: $H_s = H/2^L$ and $W_s = W/2^L$. Furthermore the dimensions of the level

in the two-dimensional heap, where (x, y) resides are

$$\begin{aligned} W_{x,y} &= W_s 2^{\lfloor \log_2(\max\{1, \frac{y}{H_s}\}) \rfloor} \\ H_{x,y} &= H_s 2^{\lfloor \log_2(\max\{1, \frac{y}{H_s}\}) \rfloor} \end{aligned} \tag{3}$$

Now the heap elements $h_{x,y}$ can be defined with the functions $h_t(x, y)$, $h_c(x, y)$ and $h_s(x, y)$:

$$\begin{aligned} h_t(x,y) &= \max\{h_{x,y+H_s}, \lfloor \log_2(|c_{x,y}|) \rfloor + 1\} \\ h_c(x,y) &= \max\{h_{2x,2y}, h_{2x+1,2y}, h_{2x,2y+1}, h_{2x+1,2y+1}\} \\ h_s(x,y) &= \lfloor \log_2(\max\{|c_{x,y}|, |c_{x+W_{x,y},y}|, |c_{x+W_{x,y},y-H_{x,y}}|\}) \rfloor + 1 \\ h_{x,y} &= \begin{cases} h_t(x,y), & \text{if } x < W_s \wedge y < H_s \\ h_s(x,y), & \text{if } x \geq W/2 \wedge y \geq H/2 \\ \max\{h_s(x,y), h_c(x,y)\}, & \text{otherwise,} \end{cases} \end{aligned} \tag{4}$$

Note that the definitions (3) and (4) are only valid for the elements in the heap, where $0 \leq y < H$ and $0 \leq x < W_s 2^{\lfloor \log_2(\max\{1, \frac{y}{H_s}\}) \rfloor}$. While the definition of the heap looks complex, we can construct the heap with a very simple and fast algorithm (Alg. 1).

2.6 Coding algorithm

The skeleton of the compression algorithm (Alg. 2) is straightforward: 1) the spatial domain image is transformed to wavelet domain by constructing the octave band composition, 2) the two dimensional heap is constructed (Alg. 1), 3) the QM-coder is initialized, 4) the coefficient matrix is scanned in bit-levels by executing scanning algorithm (Alg. 3) for each top level coefficient on each bit-level.

The decoding algorithm is similar. First an empty two dimensional heap is created by filling it with zeroes. Then the QM-decoder is initialized and the same scanning algorithm is executed in such way that instead of calculating the decisions, it extracts the decision information from the coded data.

The scanning algorithm (Alg. 3) is the core of the compression scheme. It tries to minimize correlations between the saved bits by coding as many bits as possible with zerotrees. In the pseudo-code, Bit(x,n) returns n:th bit of the absolute value of x and $s_{i,j}$ denotes the sign of the coefficient in the matrix element (i, j). Bits are coded with function QMCode(b,CONTEXT), where b is the bit to be coded and CONTEXT is the context used as explained in Section 2.8. Context can be either a constant or some function of variables known to both the coder and decoder. In both cases, the value of the context is not important, but it should be unique for each combination of parameters. Stopping of the algorithm is queried with function ContinueCoding(), which returns true, if the coding should be continued. In order to calculate the stopping condition, the quality of the approximated resulting image must be calculated while coding. This is achieved by calling function DLUpdate(n,x) every time after coding n:th bit of the coefficient x. Both calculations are explained in the Section 2.7. The dimensions of the matrix and its levels are noted in the same way as in the Section 2.5.

The scanning algorithm first checks the stopping condition. Then we check from the two dimensional heap, whether there is a zerotree starting from this location, and code the result. If the coefficient had become significant earlier, the decoder knows also that and thus we can omit the coding of the result. If we are coding a scaling coefficient ($l = 0$), we only process that coefficient and then recursively scan the coefficient in the same location on the next level below this one. If we are coding a coefficient below the top-level, we must process all three coefficients in three orientation pyramids in this spatial location and then recursively scan all four coefficients on the next level, if that level exists.

When a coefficient is processed using ScanCoeff algorithm (Alg. 4), we first check, whether it had become significant earlier. If that is the case, we just code the bit on the current bit-level and then do the distortion calculation. If the coefficient is smaller than 2^n, we code the bit on the current bit-level, and also check whether that was the first 1-bit of the coefficient. If that is true, we also code the sign of the coefficient and do the distortion calculation.

2.7 Stopping condition and distortion limiting

The DLWIC continues coding until some of the following conditions occur: 1) All the bits of the coefficient matrix have been coded, 2) The number of bits produced by the QM-coder reach a user specified threshold, or 3) the distortion of the output image, that can be constructed from sent data, decreases below the user specified threshold. The binary stopping decisions made before coding each bit of a coefficient are coded, as the decoder must exactly know when to stop decoding.

The first condition is trivial, as the main loop (Alg. 2) ends when all the bits have been coded. The second condition is also easy to implement: output routine of the QM-coder can easily count the number of bits or bytes produced. To check the third condition, the algorithm must know the MSE of the decompressed image. The MSE of the decompressed image could be calculated by doing inverse wavelet transform for the whole coefficient matrix and then calculating the MSE from the result. Unfortunately this would be extremely slow, because the algorithm must check the stopping condition very often.

The reason for using Daubechies wavelet transform is its orthonormality. For orthonormal transforms, the square sums of the pixel values of the image before and after the transform are equal:

$$\sum_{i,j}(x_{i,j})^2 = \sum_{i,j}(c_{i,j})^2 \tag{5}$$

where $x_{i,j}$ stands for the spatial domain image intensity and $c_{i,j}$ is the wavelet coefficient. Furthermore, the mean square error between the original image and some approximation of it can be calculated equally in the wavelet and spatial domains. Thus we do not have to do the inverse wavelet transform to calculate the MSE.

Instead of tracking the MSE, we track the current square error, *cse*, of the approximated image because it is computationally easier. The initial approximation

of the image is zero coefficient matrix, as we have to approximate the coefficients to be zero, when we do not know their signs. Thus the initial *cse* equals to the energy of the coefficient matrix.

$$cse \leftarrow \sum_{i,j} (c_{i,j})^2 \tag{6}$$

After sending each bit of a coefficient c, we must update *cse* by subtracting the error produced by the previous approximation of c and adding the error of its new approximation. The error of an approximation of c depends only on the level of the last known bit and the coefficient c itself. If we code the n:th bit of c, then the *cse* should be updated:

$$cse \leftarrow cse - \begin{cases} \left[(|c|\mathrm{AND}_2(2^n - 1)) - 2^{n-1}\right]^2 - \\ \left[(|c|\mathrm{AND}_2(2^{n-1} - 1)) - 2^{n-2}\right]^2 & \text{if} \lfloor \log_2 c \rfloor > n - 1 \\ c^2 - (2^{(n-1)} + 2^{(n-2)} - c)^2 & \text{if} \lfloor \log_2 c \rfloor = n - 1 \\ 0 & \text{if} \lfloor \log_2 c \rfloor < n - 1, \end{cases} \tag{7}$$

where AND_2 is bitwise and-operation. The first case defines the error reduced by finding out one bit of a coefficient, when the sign is already known. The second case defines the error reduced by finding out the sign of a coefficient and the last case states that *cse* does not change if only zero bit before the coefficients first one bit is found. The equation 7 holds only, when $n > 1$.

2.8 The use of contexts in QM-coder

The QM-coder is a binary arithmetic coding algorithm that tries to code binary data following some probability distribution as efficiently as possible. Theoretically an arithmetic coder compresses data according to its entropy [4], but the QM-coder uses a dynamical probability estimation technique [6, 7, 2] based on state automata, and its compression performance can even exceed the entropy, if the local probability distribution differs from the global distribution used in the entropy calculation.

The DLWIC codes different types of information with differing probability distributions. For example the signs of coefficients are highly random, that is the probability of plus sign is approximately 0.5, but the probability of finding the stopping condition is only $1/N$, where N is the number of stopping condition evaluations. If bits following the both distributions would be coded using the same probability distribution, the compression performance obviously would not be acceptable.

To achieve better compression performance the DLWIC uses separate *contexts* for binary data following different probability distributions. The contexts for coding the following type of data are defined: 1) signs, 2) stopping conditions, 3) the bits of the coefficients after the first one bit, 4) the bits of the scaling coefficients, 5) zerotrees on the different levels of the pyramid, 6) the significance check of the

Figure 4: Test images from top left: 1) barb (512 × 512), 2) bird (256 × 256), 3) boat (512 × 512), 4) bridge (256 × 256), 5) camera (256 × 256), 6) circles (256 × 256), 7) crosses (256 × 256), 8) france (672 × 496) and 9) frog (621 × 498).

insignificant coefficients on different pyramid levels on different orientation pyramids. The number of separate contexts is $4 * (l + 1)$, where l defines the number of levels in the pyramids. It would also be possible to define different contexts for each bit-level, but dynamical probability estimation in the QM-coder seems to be so efficient that this is not necessary.

3 Test results

The performance of the DLWIC algorithm is compared to the JPEG and the vqSPIHT [3] algorithms with a set (Fig. 4) of 8 bit grayscale test images. The vqSPIHT is an efficient implementation of the SPIHT [8] compression algorithm. The vqSPIHT algorithm uses the biorthogonal B97 wavelet transform [10], the QM-coder and a more complicated image scanning algorithm than the DLWIC. Image quality is measured in terms of *peak signal to noise ratio* [1] (PSNR), which is an inverse logarithmic measure calculated from MSE.

3.1 Compression efficiency

To compare the compression performance of the algorithms, the test image set is compressed with different BPP-rates from 0.1 to 3.0 and the PSNR is calculated as the mean for all the images. Because it is not possible to specify BPP as a parameter for JPEG compression algorithm, various quantization parameters are used and the BPP value is calculated as a mean value of the image set for each quantization value.

As can be seen in the Figure 5, the performance of the vqSPIHT and the DLWIC algorithms is highly similar. This is somewhat surprising, because of the greater complexity and better wavelet transform used in the vqSPIHT. The quality of the images compressed with the EZW variants seem to exceed the quality produced by

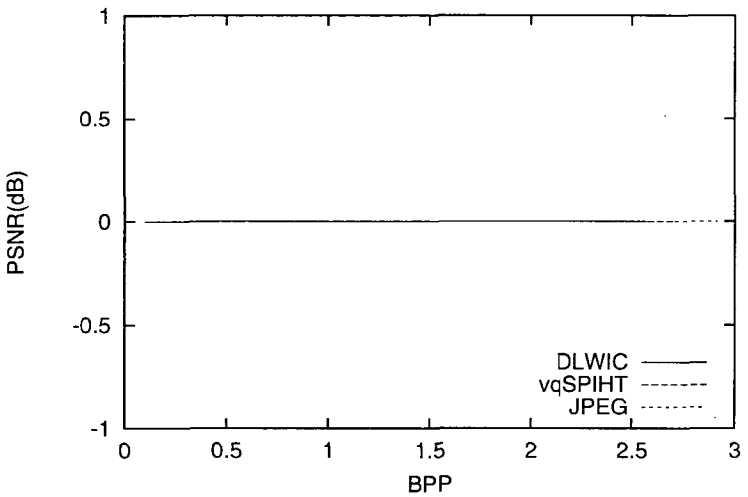


Figure 5: Compression performance comparison of DLWIC, vqSPIHT and JPEG. The PSNR-values correspond to mean value obtained from test image set (Fig. 4).

the JPEG. This is especially true when low bit-rates are used. Poor scalability of the JPEG to low BPP values is probably implied by the fixed block size used in the DCT transform of the JPEG as opposed to multi-resolution approach of the wavelet based methods.

One might expect that a conventional image compression algorithm such as the JPEG would give similar PSNR and BPP values for similar images when fixed quantization parameter is used. This is not the case as demonstrated in the Figure 6, where all the test images are compressed using the same quantization parameter (20) with the standard JPEG.

3.2 Speed and memory usage

The speed of the implementation is not compared to other techniques, because the implementation of the algorithm is not highly optimized. Instead an example of the time consumption of the different components of the compression process is examined using the GNU profiler. The frog test image is compressed using a 400MHz Intel Pentium II workstation running Linux and the algorithm is implemented in C language and compiled with GNU C 2.7.2.1 using "-O4 -p" options. The cumulative CPU time used in the different parts of the algorithm is shown in the Figure 7.

When the image is compressed with a low BPP-rate, most of the time is consumed by the wavelet transform. When the BPP-rate increases, the time used by the QM-coder, the scanning algorithm and the distortion calculations increases in

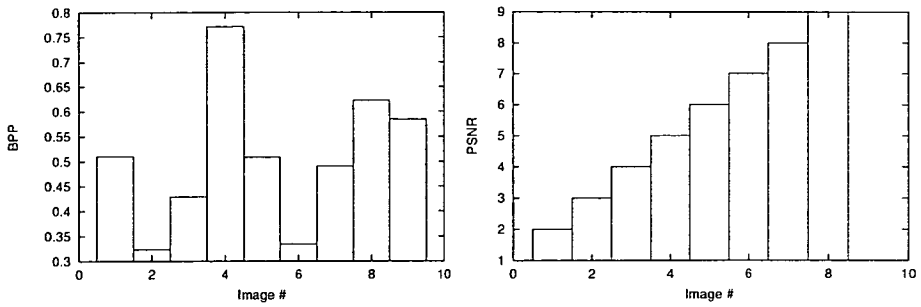


Figure 6: All the images of the test image set are compressed by JPEG with the same quantization value (20), and the BPP (left) and the PSNR (right) of the resulting images are shown.

somewhat linear manner. Construction of the two dimensional heap seems to be quite fast operation and distortion limiting is not very time consuming.

If we want to optimize the implementation of the DLWIC, the biggest problem would probably be the extensive use of the QM-coder, that is already highly optimized. One way to alleviate the problem would be to store the stopping condition is some other way than compressing the binary decision after each bit received. Also the transform would have to be optimized to achieve faster compression, because it consumes nearly half of the processing time, when higher compression ratios are used.

Probably the biggest advantage of the DLWIC over the SPIHT and even the vqSPIHT is its low auxiliary memory usage. The only auxiliary data-structure used, the two dimensional heap, can be represented using 8-bit integers and thus only consumes approximately $8 * N/3$ bits of memory, where N is the number of coefficients. If the coefficients are stored with 32-bit integers, this implies 8% auxiliary memory overhead, which is very reasonable, when compared to 32% overhead in the vqSPIHT or even much higher overhead in the SPIHT algorithm, which depends on the target BPP-rate.

4 Summary and conclusion

In this paper a new general purpose wavelet image compression scheme, DLWIC, was introduced. Also it was shown how the distortion of the resulting decompressed image can be calculated while compressing the image and thus how the distortion of the compressed image can be limited. The scanning algorithm in the DLWIC is very simple and it was shown, how it can be efficiently implemented using a two dimensional heap structure.

Compression performance of the DLWIC was tested with a set of images and the compression performance seems to be promising, when compared to a more complex

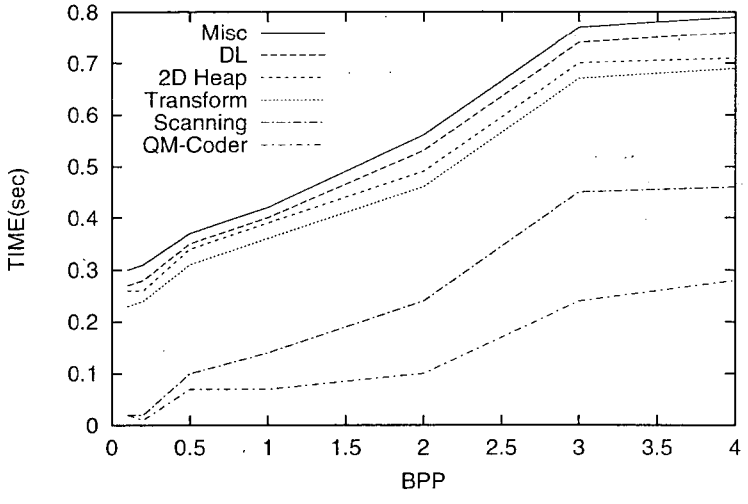


Figure 7: Running time of the different components in the DLWIC compression/decompression algorithm, when compressing the frog test image (Fig. 4). Graph shows the cumulative CPU time consumption when different BPP-rates are used.

compression algorithm, the vqSPIHT. Furthermore, the compression performance easily exceeds the performance of the JPEG, especially when high compression ratios are used.

Further research for extending the DLWIC algorithm to be used in lossless or nearly lossless multidimensional medical image compression is planned. Also the implementation of the DLWIC will be optimized and usage of some other wavelet transforms will be considered.

References

[1] R. Gonzalez and R. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, 1992.

[2] ITU-T. Progressive bi-level image compression, recommendation t.82. Technical report, International telecommunication union, 1993.

[3] A. Järvi, J. Lehtinen, and O. Nevalainen. Variable quality image compression system based on SPIHT. *to appear in Signal Processing: Image Communications*, 1998.

[4] Sayhood K. *Introduction to Data Compression.* Morgan Kaufmann, 1996.

[5] Juha Kivijrvi, Tiina Ojala, Timo Kaukoranta, Attila Kuba, László Nyúl, and Olli Nevalainen. The comparison of lossless compression methods in the case of a medical image database. Technical Report 171, Turku Centre for Computer Science, April 1998.

[6] W.B. Pennebaker and J.L. Mitchell. Probability estimation for the q-coder. *IBM Journal of Research and Development*, 32(6):737–752, 1988.

[7] William Pennebaker and Joan Mitchell. *Jpeg : Still Image Data Compression Standard.* Van Nostrand Reinhold, 1992.

[8] Amir Said and William A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, June 1996.

[9] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. Signal Processing*, 31(12), December 1993.

[10] M. Vetterli and J. Kovačević. *Wavelets and Subband Coding.* Prentice Hall, Englewood Cliffs, NJ, 1995.

Algorithm 1 Construct2DHeap

for $l \leftarrow 1$ **to** $L+1$ **do**
  $H_t \leftarrow H/2^{\min\{l,L\}}$, $W_t \leftarrow W/2^{\min\{l,L\}}$
  for $j \leftarrow 0$ **to** $H_t - 1$ **do**
    for $i \leftarrow 0$ **to** $W_t - 1$ **do**
      if l = 1 **then**
        $t \leftarrow 0$
        $u \leftarrow \lfloor 1 + \log_2(\max\{|c_{i,j+H}|, |c_{i+W,j}|, |c_{i+W,j+H}|\})\rfloor$
      else if $l \leq L$ **then**
        $t \leftarrow \max\{h_{2x,2y}, h_{2x+1,2y}, h_{2x,2y+1}, h_{2x+1,2y+1}\}$
        $u \leftarrow \lfloor 1 + \log_2(\max\{|c_{i,j+H}|, |c_{i+W,j}|, |c_{i+W,j+H}|\})\rfloor$
      else
        $t \leftarrow \max\{h_{i,j+H_s}, h_{i+W_s,j}, h_{i+W_s,j+H_s}\}$
        $u \leftarrow \lfloor 1 + \log_2(\max\{(|c_{i,j}|)|0 \leq i < W \wedge 0 \leq j < H\})\rfloor$
      $h_{i,j} \leftarrow \max\{t, u\}$

Algorithm 2 CompressDLWIC

Transform the spatial image with Daubechies wavelet transform constructing the octave band composition where the coefficients $c_{i,j}$ are represented with positive integers and separate sign bit.
Construct the two dimensional heap (Alg. 1).
Initialize QM-coder
Calculate initial distortion of the image (Section 2.7).
$n_{max} \leftarrow \max\{h_{i,j}|0 \leq i < W_s \wedge 0 \leq j < H_s\}$
for $n \leftarrow n_{max}$ **to** 1 **do**
  for $j \leftarrow 0$ **to** $H_s - 1$ **do**
    for $i \leftarrow 0$ **to** $W_s - 1$ **do**
      Scan$(i, j, 0, n)$ (Alg. 3)

Algorithm 3 Scan(i, j, l, n)

if ContinueCoding() **then**
  if $h_{i,j} < n$ **then**
    QMCode(INSIGNIFICANT, SIGNIFICANCE-TEST(l))
  else
    if $h_{i,j} = n$ **then**
      QMCode(SIGNIFICANT, SIGNIFICANCE-TEST(l))
    if $l = 0$ **then**
      ScanCoeff$(i, j$,TOPLEVEL$, n)$
      Scan$(i, j + H_s, 1, n)$
    else
      ScanCoeff$(i, j$,HORIZONTAL$\left(\begin{array}{l} l, c_{(i+W_{i,j}),j} < 2^n, \\ c_{(i+W_{i,j}),(j-Hi,j)} < 2^n \end{array} \right), n)$
      ScanCoeff$(i + W_{i,j}, j$,DIAGONAL$\left(\begin{array}{l} l, c_{i,j} < 2^{n-1}, \\ c_{(i+W_{i,j}),(j-Hi,j)} < 2^n \end{array} \right), n)$
      ScanCoeff$(i + W_{i,j}, j - H_{i,j}$,VERTICAL$\left(\begin{array}{l} l, c_{i,j} < 2^{n-1}, \\ c_{(i+W_{i,j}),j} < 2^{n-1} \end{array} \right), n)$
      if $2 * y < H$ **then**
        Scan$(2 * i, 2 * j, l + 1, n)$
        Scan$(2 * i + 1, 2 * j, l + 1, n)$
        Scan$(2 * i, 2 * j + 1, l + 1, n)$
        Scan$(2 * i + 1, 2 * j + 1q, l + 1, n)$

Algorithm 4 ScanCoeff$(x, y$,CONTEXT$, n)$

if $c_{x,y} < 2^n$ **then**
  QMCode(Bit$(c_{x,y}, n)$,CONTEXT)
  if Bit$(c_{x,y}, n) = 1$ **then**
    QMCode($s_{x,y}$,SIGN)
    DLUpdate$(n, c_{x,y})$
else
  QMCode(Bit$(c_{x,y}, n)$,COEFFICIENTBIT)
  DLUpdate$(n, c_{x,y})$

Acta Cybernetica **14** (1999) 357-376.

On the Exact Solution of the Euclidean Three-Matching Problem

Gábor Magyar Mika Johnsson Olli Nevalainen *

Abstract

Three-Matching Problem (3MP) is an NP-complete graph problem which has applications in the field of inserting electronic components on a printed circuit board. In 3MP we want to partition a set of $n = 3l$ points into l disjoint subsets, each containing three points (triplets) so that the total cost of the triplets is minimal. We consider the problem where the cost c_{ijk} of a triplet is the sum of the lengths of the two shortest edges of the triangle (i, j, k); the reason for this assumption is the nature of the practical problems.

In this paper we discuss the optimal solution of 3MP. We give two different integer formulations and several lower bounds of the problem based on the Lagrangian relaxations of the integer programs. The different lower bounds are evaluated by empirical comparisons. We construct branch-and-bound procedures for solving 3MP by completing the best lower bound with appropriate branching operations. The resulting procedures are compared to our previous exact method and to general MIP solvers.

1 Introduction

The task in Three-Matching Problem (3MP) is to form l disjoint triplets from $n = 3l$ points and to minimize the total cost of these triplets, i.e., to connect the three points of each triplet by two line segments so that the total length of the line segments is minimal. The 3MP can be illustrated with an Euclidean problem instance, see Fig. 1.

The 3MP occurs in some industrial applications [3, 5, 7, 12]. In manual insertion of electronic components on a printed circuit board, the operations are arranged into close triplets to aid the worker's task. Furthermore, some flexible machines (e.g., General Surface Mounter) for automatic electronic component insertion have from three to eight insertion heads and operate in cycles comprising component pickups and insertions; the throughput of the machine can be improved by minimizing the length of the inter-board head movements. Component insertions are performed in two phases: in the first phase head nozzles pick up new components from the component feeders, and in the second phase the head moves to the actual insertion

*Turku Centre for Computer Science (TUCS) and Department of Computer Science, University of Turku, Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland

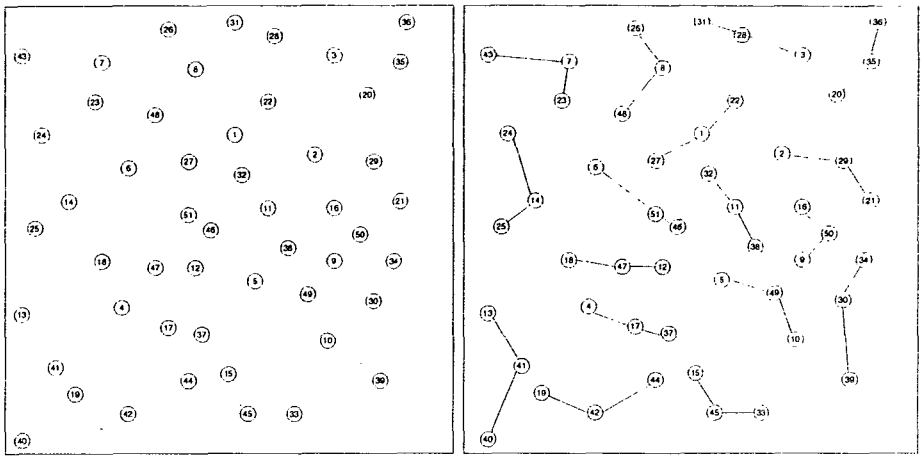


Figure 1: A sample 51-point 3MP problem instance ('eil51' from TSPLIB [15]) along with its optimal solution. The vertices are numbered as they appear in the problem file.

points to perform the actual insertions. Careful selection of the point triplets (in the case of a three-headed machine) is essential in minimizing the total printing time of the board. A similar problem is encountered in the scheduling of an automated assay analysis instrument (AutoDelphia).

3MP is closely related to the 3-dimensional assignment problem (3DA) [1, 3] which is NP-complete and a restricted version of our problem (in 3DA the points of the triplets are drawn from three disjoint subsets of the point set). Our previous studies and the connection to 3DA indicate that 3MP is NP-complete [6]. Additionally, 3MP is closely related to the p-median problem [2], in which we search p median points to which the remaining points are connected in such a way that the total sum of the connecting edges is minimal. However, the p-median problem does not restrict the number of allocated points to a median to be exactly n/p. The 3MP can therefore be viewed as a specialization of the p-median problem.

The 3MP can be solved heuristically by standard approaches, like local search heuristics using pairwise interchanges, simulated annealing, tabu search and genetic algorithm [6, 9, 10]. Several lower bounds can be given to the problem. By knowing the optimal solution to the problem we can evaluate the quality of the lower bound and the upper bound solutions of the heuristic approaches. Here we will discuss a number of design alternatives of B&B algorithms for 3MP and their trade-offs in this study.

The plan of the paper is as follows. Two different integer programming formulations for 3MP are given in the next section. In Section 3 we discuss briefly the branch-and-bound method and its main components. Section 4 describes four lower bounds to the problem and compares them empirically. The best lower bound is

completed with three different branching rules in Section 5. Test results comparing the branch-and-bound procedures and previous methods are discussed in Section 6. This section also compares our procedure to general MIP solver packages. Finally, the conclusions are drawn in Section 7.

2 Problem formulations

In this section, we give two integer programming formulations for 3MP. Firstly, the problem can be formulated by the following 0-1 program [6]:

Let d_{ij} represent the cost of the edge $j \to i$, V the set of vertices and x_{ij} the decision variable where

$$x_{ij} = \begin{cases} 1 & \text{if the edge } j \to i \text{ is present in the solution,} \\ 0 & \text{otherwise.} \end{cases}$$

The problem is to

$$\min z_{IP1} = \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_{i \in V} x_{ij} + \frac{1}{2} \sum_{k \in V} x_{jk} = 1 \qquad \forall j \in V \tag{2}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V, j \in V \tag{3}$$

In the above formulation the edges are directed and the first index stands for the central point of the triplets. The first term in constraint (2) is equal to 1 if and only if the point j is not a central triplet point, whereas the second sum is equal to 1 if and only if the point j is a central point.

Our second formulation considers 3MP as a modification of the p-median problem (e.g., see [2]), where $p = n/3$ and every median has exactly two other points allocated to it. The decision variable x_{ii} is equal to 1 if and only if point i is a median vertex. Concerning the other variables,

$$x_{ij} = \begin{cases} 1 & \text{if point } j \text{ is allocated to median } i \text{ in the solution,} \\ 0 & \text{otherwise.} \end{cases}$$

The problem is to

$$\min z_{IP2} = \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} \tag{4}$$

subject to

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \tag{5}$$

$$\sum_{i \in V} x_{ii} = n/3 \tag{6}$$

$$\sum_{j \in V \& j \neq i} x_{ij} = 2x_{ii} \qquad \forall i \in V \tag{7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V, j \in V \tag{8}$$

Equations (5) ensure that each point is allocated to a median. Equation (6) says that there are exactly $n/3$ medians. Equations (7) ensure that every point is allocated only to a median point and every median has exactly two other points allocated to itself. The above program is also valid for the 3MP if we have inequalities in (7).

When we describe the Lagrangian relaxations of the above integer programs in Section 4, we will refer to the first program as "IP1", the second one with signs "=" in (7) as "IP2a", and with signs "$\leq$" as "IP2b".

It must be noted that the p-median formulation does not hold for natural generalizations of the three-matching problem: for example, in the case of 4-matching when the objective is to find point groups of four points with a minimal total distance of the path crossing the four points, there is no median point.

3 Outline of the algorithm

The *branch-and-bound* (B&B) method (e.g., see [8]) is an intelligent enumerating procedure for finding the exact solution of a given combinatorial optimization problem. The method maintains a set of live leaves of an enumeration tree. The tree represents partitions of the feasible solutions in its nodes. The classes of the partitions are usually defined by storing some fixed variables or partial solutions in the nodes. The procedure begins with the root, which has no fixed variables; it represents all the feasible solutions of the optimization problem under consideration. The *branching* operation is used to partition the feasible solutions of a selected branching node into its descendant nodes. The partitioning is done by fixing some further variables in the descendant nodes in a systematic way depending on the structure of the problem. The effectiveness of the method comes from the *bounding* operation, where we calculate sharp lower bounds (in case of a minimization problem) of the object values of the feasible solutions represented in the nodes of the tree. We do not have to build up the whole enumeration tree in general, i.e., if a node does not contain any better solutions than a known one, then that node and the respective subtree can be discarded. A more precise description of the branch-and-bound procedure is as follows:

1. *(Initialization)* Let the set L of live leaves contain only one node, the root, which represents all the feasible solutions. Calculate the lower bound for the

root. Calculate a sub-optimal solution s^* with some *heuristics* if possible and store its value to z^*.

2. *(Iteration)* If L is empty, then terminate. Return s^* as the optimal solution and z^* as the optimal value.

3. *(Selection)* Select a branching node $Node$ from L with some leaf selection rule.

4. *(Branching)* Form the partition $\varphi(Node) = \{N_1, \ldots, N_k\}$ of the selected branching node.

5. *(Bounding)* Calculate the lower bounds $g(N_i)$ for all the new nodes, $i = 1, \ldots, k$. If a feasible solution arises during the bound calculations, then modify the current best solution s^* and its object value z^* when necessary.

6. *(Fathoming)* Update the set L of live leaves according to the explored node and the actual value of z^*:

   (a) add $N_1, \ldots, N_k$ to L,

   (b) delete $Node$ from L,

   (c) delete all $N \in L$ for which $g(N) \geq z^*$.

7. *(Next iteration)* Go to the next iteration (step 2).

We have two major alternatives to select a leaf at step 3. We can select the leaf with the minimal assigned lower bound. This way the set of live leaves usually grows very rapidly since we build the tree more or less level by level horizontally. A more practical approach selects the leaf for which the ratio of the assigned lower bound and the number of fixed variables is minimal. In this case we traverse the tree in a depth-first fashion. This action provides feasible solutions at an early phase of the processing and uses less memory. We will apply the second selection criterion in our branch-and-bound procedures.

The main components of a B&B procedure are the branching operation φ, the bounding function g, and the primal heuristics available for generating initial solutions. Furthermore, upper bound heuristics can also be applied to generate candidate solutions based on the partial solutions of the tree nodes. Next we consider all these components in detail and give a complete branch-and-bound procedure.

4 Lower bounds

In this section we discuss four lower bounds for the 3MP. The first bound is problem-specific, while the other three are based on the Lagrangian relaxations of IP1 and IP2a. The latter program provides two alternatives for the relaxation, either by relaxing the constraints (5) or (7).

The first lower bound ("LB1"), introduced in [6] (as bound "D") is applicable for the Euclidean case only:

1. For each point, calculate the distance to the closest point, store these distances together with the indices in the list S_1.

2. For each point, calculate the distance to the second closest point, store these distances together with the indices in the list S_2.

3. Sort lists S_1 and S_2 into an increasing order on the distances.

4. Omit duplicates from S_1 and S_2 (i.e., rule out the edge $b-a$ if $a-b$ has occurred previously), also rule out $a-c$ if $a-b$ and $b-c$ are already in the list and $a-c$ is the longest distance in the triplet $(a-b-c)$. The deletion is done first in S_2 and in S_1 only if S_1 itself requires the deletion (all deletion conditions are met in S_1). When selecting the distances from the sorted lists S_1 and S_2, we omit those distances that would connect a point to more than two other points.

5. If there are less than $2/3n$ values in S_1, move $2/3n - |S_1|$ elements with the shortest distances from S_2 to S_1. Alternatively, if $|S_1| > 2/3n$, delete at a maximum $|S_1| - 2/3n$ values from S_1. Deletion is allowed only if the points connected by the deleted edge are still connected after the deletion to some other point in S_1 by an edge. Deletions are started from the end of S_1 (the largest values first). The operation is implemented by maintaining a list $R = (r_1, \ldots, r_{|S_1|})$ where r_i is the degree of point i in S_1. So the deletion of the element (d_{ij}, i, j) is allowed only if $r_i > 1$ and $r_j > 1$. After a successful deletion, R is updated, $r_i = r_i - 1, r_j = r_j - 1$.

6. Sum the first $2/3n$ distances in S_1 to get the lower bound.

The Lagrangian relaxation (see e.g. [4, 14]) is a general way for obtaining high quality lower bounds for hard combinatorial optimization problems. By attaching Lagrangian multipliers to some of the constraints of the original problem and relaxing these constraints into the objective function, we get a Lagrangian relaxation, which is easier to solve than the original problem. Maximizing the optimal values of the Lagrangian relaxation for λ, we obtain the Lagrangian dual program. The optimal value of the Lagrangian dual is a valid lower bound (in the case of minimization) and in the optimal case it can reach the optimal value of the linear programming relaxation of the original problem. The main advantage of the Lagrangian relaxation is that the solution process is much faster than solving the LP relaxation. The maximization of the value of the Lagrangian dual is often done by the subgradient optimization method, in which we update the Lagrangian multipliers in a systematic way to achieve the best lower bound. The Lagrangian relaxation technique has been applied successfully to many hard combinatorial optimization problems, see for example, [1, 2, 4].

Next we describe three Lagrangian lower bounds and the subgradient optimization procedure for maximizing the bounds. Our first bound is based on the Lagrangian relaxation of IP1 [6]. The latter two relaxations were introduced in [2] for the general p-median problem. Here we recall them with such modifications that make them applicable to the 3MP.

In the case of the relaxation for the problem formulation of IP1, we relax the constraints (2) into the objective function by introducing the Lagrangian multipliers $\lambda_j (j \in V)$ and get the following Lagrangian relaxation:

$$\begin{aligned} \min z_{D1} &= \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} + \sum_{j \in V} \lambda_j \left(\sum_{i \in V} x_{ij} + \frac{1}{2} \sum_{k \in V} x_{jk} - 1 \right) = \\ &= \sum_{i \in V} \sum_{j \in V} \left(d_{ij} + \lambda_j + \frac{1}{2} \lambda_i \right) x_{ij} - \sum_{j \in V} \lambda_j \end{aligned} \tag{9}$$

subject to

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V, j \in V \tag{10}$$

This program is easy to solve. The solution becomes straightforward after examining the signs of the terms $D_{ij} = d_{ij} + \lambda_j + 0.5\lambda_i$ since the second sum of (9) is constant with respect to x. Hence, the optimal solution of z_{D1} is:

$$x_{ij}^* = \begin{cases} 0 & \text{if } D_{ij} > 0, \\ 1 & \text{if } D_{ij} \leq 0. \end{cases} \tag{11}$$

Now we describe two Lagrangian relaxations for the program IP2a. Let K_1 denote the set of vertices that have been previously fixed to be medians and K_0 the vertices that have been fixed to be non-medians by the branching rules of the B&B procedure.

As a second relaxation, we relax the constraints (7) by introducing the multipliers $\lambda_i (i \in V)$. The resulting program allows us to allocate the points to non-medians and, in addition, a point may have an arbitrary number of other points allocated to itself. We get the following Lagrangian relaxation:

$$\begin{aligned} \min z_{D2} &= \sum_{i \in V - K_0} \sum_{j \in V - K_1} d_{ij} x_{ij} + \sum_{i \in V - K_1} \lambda_i \left(\sum_{j \in V \& j \neq i} x_{ij} - 2x_{ii} \right) = \\ &= \sum_{i \in V - K_0} \sum_{j \in V - K_1 \& j \neq i} (d_{ij} + \lambda_i) x_{ij} + \sum_{i \in V - K_0} (d_{ii} - 2\lambda_i) x_{ii} \end{aligned} \tag{12}$$

subject to

$$\sum_{i \in V-K_0-K_1} x_{ij} = 1 \qquad \forall j \in V - K_1 \tag{13}$$

$$\sum_{i \in V-K_0-K_1} x_{ii} = n/3 - |K_1| \tag{14}$$

$$\lambda_i = 0 \qquad \forall i \in K_1 \tag{15}$$

$$x_{ii} = 0 \qquad \forall i \in K_0 \tag{16}$$

$$x_{ii} = 1 \qquad \forall i \in K_1 \tag{17}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in V, j \in V \tag{18}$$

To solve this program, we define α_j as the minimum cost of allocation vertex j other than to itself, i.e., the minimal arising cost if vertex j is not a median:

$$\alpha_j = \min_{i \in V-K_0 \& i \neq j} (d_{ij} + \lambda_i) \qquad \forall j \in V - K_1 \tag{19}$$

The Lagrangian relaxation (equations (12)-(18)) then can be changed for the following problem which has the same optimal value as the Lagrangian relaxation. The basis of this substitution is that the minimal value of $\sum_{i \in V-K_0, i \neq j} (d_{ij} + \lambda_i) x_{ij}$ is equal to α_j if $j \in V - K_1$, by (13).

$$\min z_{D2} = \sum_{j \in V-K_1} \alpha_j + \sum_{j \in K_1} d_{jj} + \sum_{j \in V-K_0-K_1} (d_{jj} - 2\lambda_j - \alpha_j) x_{jj} \tag{20}$$

subject to

$$\sum_{i \in V-K_0-K_1} x_{ii} = n/3 - |K_1| \tag{21}$$

$$x_{ii} = 1 \qquad \forall i \in K_1 \tag{22}$$

$$x_{ii} \in \{0,1\} \qquad \forall i \in V - K_0 \tag{23}$$

The optimal solution of this program is found by setting x_{ii} to 1 for $i \in K_1$ and the remaining $n/3 - |K_1|$ x_{ii} with the smallest $(d_{ii} - 2\lambda_i - \alpha_i)$ $(i \in V - K_0 - K_1)$ to 1, while all the other x_{ii} are set to zero. If the values of the variables x_{ii} are denoted by x^*_{ii} the optimal solution of the above program, then the other variables (x^*_{ij}) are calculated as follows:

$$x^*_{ij} = \begin{cases} 1 & \text{if } x^*_{jj} = 0 \text{ and } i \text{ corresponds to the minimum} \\ & \text{for } \alpha_j \text{ in (19) } (i \neq j), \\ 0 & \text{otherwise } (i \neq j). \end{cases} \tag{24}$$

Thirdly, we relax the constraints (5) of IP2a by introducing the Lagrangian multipliers $\lambda_j (j \in V)$ and get the following Lagrangian relaxation:

$$\begin{aligned} \min z_{D3} &= \sum_{i \in V-K_0} \sum_{j \in V-K_1} d_{ij} x_{ij} - \sum_{j \in V} \lambda_j \left(\sum_{i \in V} x_{ij} - 1 \right) = \\ &= \sum_{i \in V-K_0} \sum_{j \in V-K_1} (d_{ij} - \lambda_j) x_{ij} + \sum_{j \in K_1} (d_{jj} - \lambda_j) + \sum_{j \in V} \lambda_j \end{aligned} \tag{25}$$

subject to

$$\sum_{i \in V-K_0-K_1} x_{ii} = n/3 - |K_1| \tag{26}$$

$$\sum_{j \in V-K_1 \& j \neq i} x_{ij} = 2x_{ii} \qquad \forall i \in V \tag{27}$$

$$x_{ii} = 0 \qquad \forall i \in K_0 \tag{28}$$

$$x_{ii} = 1 \qquad \forall i \in K_1 \tag{29}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in V, j \in V \tag{30}$$

The relaxation disregards the constraint that every point must be allocated to a median. Therefore, every selected median point will still have exactly two other points allocated to it, but a point can be allocated to several medians.

To solve this program, let us consider the effect of specifying that k is a median vertex. This implies the settings $x_{kk} = 1$ and $x_{kj} = 1$ where j corresponds to the indices of the smallest or second smallest value of $(d_{kj} - \lambda_j)$, $j \in V - K_1 - \{k\}$. This is true, because the constraints (27) ensure that every selected median will have the two points with the smallest $(d_{kj} - \lambda_j)$ allocated to itself. Because the constraints (5) have been relaxed, a point can be allocated to several medians. We denote by α_k the arising cost when deciding that vertex k will be a median:

$$\alpha_k = (d_{kk} - \lambda_k) + \min_{j \in V-K_1 \& j \neq k} (d_{kj} - \lambda_k) + \operatorname{2ndmin}_{j \in V-K_1 \& j \neq k} (d_{kj} - \lambda_k), \forall k \in V - K_0 \tag{31}$$

where "2ndmin" denotes the second smallest value of the expression.

The solution of the Lagrangian relaxation of (25)-(30) now can be obtained by solving the following problem which has the same optimal value:

$$\min z_{D3} = \sum_{i \in V-K_0} \alpha_i x_{ii} + \sum_{j \in V} \lambda_j \tag{32}$$

subject to

$$\sum_{i \in V - K_0 - K_1} x_{ii} = n/3 - |K_1| \tag{33}$$

$$x_{ii} = 1 \qquad \forall i \in K_1 \tag{34}$$

$$x_{ii} \in \{0,1\} \qquad \forall i \in V - K_0 \tag{35}$$

The optimal solution of this program is found by selecting the needed $n/3 - |K_1|$ medians as follows: set x_{ii} to 1 for the indices i ($i \in V - K_0 - K_1$) which give the smallest α_i values. The other x_{ii} values (except $i \in K_1$) are set to 0. Denoting the optimal assignment of the diagonal elements by x^*_{ii}, the remaining variables (x^*_{ij}) are allocated as follows:

$$x^*_{ij} = \begin{cases} 1 & \text{if } x^*_{ii} = 0 \text{ and } j \text{ corresponds to the smallest or the} \\ & \text{second smallest index for } \alpha_i \text{ in (31) } (i \neq j), \\ 0 & \text{otherwise } (i \neq j). \end{cases} \tag{36}$$

The problem reduction by calculating penalties is a common technique in Lagrangian relaxations. Suppose that we have solved the Lagrangian relaxation to optimality. We can then estimate the increase in the lower bound which would result from forcing a variable to either 0 or 1. If the lower bound resulting from imposing some condition on a variable is above some known upper bound to the problem, then that condition cannot be satisfied in the optimal solution of the original problem. This means that we can fix the value of the given variable as it is in the optimal solution of the Lagrangian relaxation. For example, if a variable x^*_{ij} is 0 in the optimal solution of the Lagrangian relaxation, and the penalty for forcing x^*_{ij} to be 1 results in exceeding the known upper bound, then the value of x^*_{ij} can be fixed to 0 and the size of the problem can be reduced. For the third relaxation, we include the calculation of the penalties, where we have four cases for $x^*_{ij} = 0$ and two cases for $x^*_{ij} = 1$.

Finally, we describe the method that is applied for maximizing the optimal values of the Lagrangian dual programs. We will denote the three described Lagrangian relaxations by "LR1", "LR2" and "LR3" and "LR3P" (LR3 with penalties). In the above relaxations, the Lagrangian multipliers are unconstrained in sign, because the relaxed constraints are equalities in every case. We apply the following subgradient optimization procedure in order to maximize the value of the Lagrangian dual programs:

1. Determine an upper bound z_{UB} to the problem. This can be done by any of our previous heuristic methods, see for exapmle, [6, 9, 10]. Set z^*_L to $-\infty$, which will denote the maximal lower bound.

2. Initialize the Lagrangian multipliers to $\lambda_j = 0$, $j \in V$.

3. Solve the Lagrangian relaxation with the actual set of Lagrangian multipliers. Let z^*_D denote the optimal value of the relaxation, x^*_{ij} the optimal solution and M^* the associated median set (in the case of LR2 and LR3).

4. Set $z_L^* = \max(z_D^*, z_L^*)$.
5. Terminate if $z_{UB} \leq z_L^*$, which means that the maximal lower bound is found. Otherwise go to step 6.
6. Perform the penalty tests outlined above (case LR3P).
7. Calculate the subgradient vector S:

$$S_j = \sum_{i \in V} x_{ij}^* + \frac{1}{2} \sum_{k \in V} x_{jk}^* - 1 \qquad \forall j \in V \text{for LR1} \tag{37}$$

$$S_i = \sum_{j \in V \& j \neq i} x_{ij}^* - 2x_{ii}^* \qquad \forall i \in V \text{for LR2} \tag{38}$$

$$S_j = 1 - \sum_{i \in M^*} x_{ij}^* \qquad \forall j \in V \text{for LR3} \tag{39}$$

8. If $S_j = 0$ for all $j \in V$, then terminate; the optimal solution of the original problem is found. Otherwise go to step 9.
9. Calculate the step size T for updating the multipliers by

$$T = \frac{\pi(z_{UB} - z_D^*)}{\sum_{i \in V} S_i^2} \tag{40}$$

where π is a constant $(0 < \pi \leq 2)$ controlling the step size of the procedure.

10. Update the multipliers by

$$\lambda_j = \lambda_j + TS_j, \qquad \forall j \in V \tag{41}$$

11. Go to step 3 for the next subgradient iteration unless some termination criteria (see below) is satisfied. Else stop and return z_L^* as the maximal lower bound.

We set the value of π initially to 2. After that it is halved after every 30 iterations if the value of the best lower bound has not been improving. The procedure terminates after reaching a predefined total number of iterations which is set to 200.

Now we are in the position to compare the lower bounds. Table 1 shows a summary of the results of practical tests with 20 Euclidean problem instances[1]. It should be noted that all the above bounds except the first one (LB1) can deal with

[1]The test problems are available from http://www.cs.utu.fi/research/projects/3mp/

Prob	Pts	LR3P	LR3	LR1	LR2	LB1	max LB	OPT
f21	21	0,00	0,03	8,31	8,31	7,52	159,73	159,73
f27	27	1,22	1,22	7,88	7,91	10,45	7913,53	8011,08
f33	33	0,00	0,00	7,31	7,32	8,31	255,69	255,69
f39	39	10,63	10,63	15,79	15,79	14,31	252,31	282,31
39a	39	1,56	1,56	9,98	9,98	7,55	771,74	783,96
39b	39	7,47	7,47	17,03	17,06	16,76	764,45	826,14
39c	39	0,54	0,54	9,81	9,84	11,44	954,21	959,38
39d	39	1,65	1,65	10,41	10,44	10,34	768,33	781,22
39e	39	6,25	6,25	13,00	13,01	16,11	817,82	872,35
42a	42	6,14	6,14	15,14	15,16	15,77	890,83	949,09
42b	42	5,41	5,41	15,43	15,48	15,28	814,00	860,59
45a	45	0,55	0,55	10,92	10,94	10,52	1007,60	1013,14
45b	45	4,20	4,20	20,34	20,36	22,62	944,18	985,60
48a	48	0,00	0,00	5,12	5,13	4,84	996,61	996,61
48b	48	0,00	0,00	15,52	15,53	11,66	967,83	967,83
51a	51	1,78	1,78	14,19	14,20	13,26	966,01	983,55
51b	51	3,01	3,01	11,07	11,09	9,17	973,57	1003,82
eil51	51	2,36	2,36	8,33	8,33	8,66	259,34	265,61
f99	99	0,89	0,89	5,99	5,99	6,88	382,78	386,23
rat99	99	1,07	1,07	8,88	8,89	9,60	743,48	751,53
Average		2,7367	2,7382	11,5233	11,5372	11,5542		

Table 1: Test results for different lower bounds and the effect of the penalties for the bound of LR3. The values show the difference from the optimal value in percents.

any kind of distance matrix. The problem instances "rat99" and "eil51" are from the TSPLIB [15] and the instances "f21", "f27", "f33", "f39", "f99" are prefixes of the files "eil51", "krob150", "rat99", "rat783" and "eil101" of the TSPLIB, respectively. The other 13 files are generated by picking the co-ordinates of the points randomly and independently within the range [1,300]. The results of Table 1 show the difference from the optimum. Column 8 shows the value of the maximal lower bound of the five bounds and the last column shows the optimal value, which we calculated with the B&B procedure. The averages of the differences from the optimum are indicated in the last row.

The results indicate that LR3 outperforms the other bounds. Furthermore, the penalties still improve the bound in one case (f21). The average difference between the bound LR3 (LR3P) and the optimum is 2.7367% for the test problems. Some problems (e.g., f39, 39b and 39e) are harder than the others because the bound is farther from the optimum. On the other hand, the bound can also be very close to the optimum. In some cases it even reaches the optimum, for exapmle, f21, f33, 48a and 48b.

The results of LR1 and LR2 are practically identical and LB1 also seems to be as good as these two bounds. The difference between LR3 and the three other bounds is relatively large. Therefore, the application of LR3 is justified in the B&B procedure, as the quality of the lower bound is essential. However, the calculation

of the bound LB1 is much faster than of the other three, which require many hundreds of iteration steps of the subgradient method. Therefore, we will examine the application of this bound in the B&B procedure.

Finally, the results of Table 1 are also in line with the observations of [2], i.e., the bound LR3 clearly outperforms LR2.

5 Branching rules

Next we equip LR3P with appropriate branching rules to build up a complete branch-and-bound procedure for the 3MP. We describe three different branching rules. The first one was introduced in [2] for LR3. The other two can be considered as enhancements or extensions of the first one. The branching rules choose variables to the sets of K_0 (non-medians) and K_1 (medians) (cf. Section 4) systematically. A variable is referred as a free variable unless the branching rules or some other operations, for example the application of the penalties, have fixed its value.

Rule 1. The first branching rule [2] selects the variable x_{jj} corresponding to

$$\alpha_j = \min_{i \in M^* - K_1} \alpha_i \tag{42}$$

and sets x_{jj} either to 0 (non-median, $K_0' = K_0 + \{j\}$) or to 1 (median,$K_1' = K_1 + \{j\}$) in the descendant nodes. This rule selects the median point from the selected free medians of the Lagrangian relaxation and gives a binary enumeration tree.

Rule 2. This rule performs a 3-ary branching based on a free variable x_{jk}, where j is determined by (42) as above. The vertex k corresponds to $\min d_{jk}$ among the free variables x_{jk} (i.e., it represents the smallest possible allocation cost of a point to j). In one branch, we perform the partitioning by fixing x_{jj} to 0 ($K_0' = K_0 + \{j\}$). In the other two branches, we set j as a median by fixing $x_{jj} = 1$ ($K_1' = K_1 + \{j\}$). The latter two branches differ in the value of the variable x_{jk}, namely we fix $x_{jk} = 1$ in one of them (this also yields that k becomes a non-median), and $x_{jk} = 0$ in the other. This rule is an extension of rule 1 in the sense that in addition to deciding whether point j is a median or not, we also decide an allocated point k in the case when j was selected as a median.

Rule 3. This rule is a modification of rule 2. We select now the branching variable x_{jk} corresponding to $\min d_{jk}$, where $j \in V - K_0 - K_1, k \in V - K_1, k \neq j$ and x_{jk} is free. The difference from the previous rule is that the candidate for the median point (j) is not determined by (42), but it is selected according to the minimal value of d_{jk}, where j can be any of the possible candidates for a median. The branching is similar to rule 2, this again yields a 3-ary tree.

The above branching rules are exhausted when we have fixed the required number of medians, i.e., $|K_1| = n/3$ (or $|K_0| = 2n/3$). It seems to be problematic that in the p-median problem the solution resulting from the relaxation LR3 is restricted to find the $n/3$ median points. In the general p-median problem, the

remaining points are allocated to the closest medians and a median can have an arbitrary number of other points allocated to it. The solution of 3MP, however, requires that each median has exactly two other points allocated. This difficulty can be overcome by applying the Hungarian method (e.g., see [13]) as follows: Let us create an assignment problem where we put the median points in duplicates into the first set (i.e., every selected median point will appear twice). The other set includes the non-median points. Performing the Hungarian method for the assignment of the elements of the two sets results for each median point a matching to exactly two non-median points, where the allocated points are different for each pair of median points. With this additional procedure, the solution of 3MP is complete after solving the appropriate restricted version of the general p-median problem. For this reason, it is enough to select the $n/3$ median points optimally (central points of the triplets), and the exact solution of the 3MP can be derived from that.

In order to find tight upper bounds during the B&B procedure, we perform the Hungarian method also on the median set M^* which is associated with the best Lagrangian lower bound from the subgradient procedure. This way the size of the tree can be reduced since the better upper bounds enable us to fathom some more unnecessary leaves, see step 6 of the B&B procedure in Section 3.

It was noted above that the calculation of a high quality lower bound is achieved by 200 iterations of the subgradient method. This is applied at the root node. We perform only 30 iterations at other levels of the tree. The Lagrangian multipliers of the tree nodes are initialized to their best values in the parent nodes; this common practice enables a smaller number of iterations at the lower levels of the tree. The value of π is initialized to 1 in the tree nodes, and it is halved after every fifth iteration.

When adding a variable to the set K_0, all the variables in its row are also fixed to 0, as it cannot have any points allocated. If a variable is added to K_1, the variables in its column (except the diagonal) are fixed to 0, as it cannot be allocated to other points. We apply some additional tests on the fixed variables, and, if it is possible, we still fix some more variables. These additional tests take place after the penalty tests and they work recursively until no new variable has been fixed.

The following tests are performed:

1. If all three variables have been assigned to the median (the median and the two allocated points), then all the other free variables of the row of the median can be fixed to 0.

2. If a row of a median contains only the required number of free variables (i.e., 1 or 2), then those variables are automatically allocated to the median; furthermore, the appropriate columns are included in the set of K_0. The other variables in the row and column of the automatically allocated variables are fixed to 0.

3. If a row has less than three free variables and no variable has been fixed to 1, then it is impossible for it to be a row of a median. Therefore, the

		Time (sec)			Nodes		
Problem	Pts	P1	P2	P3	P1	P2	P3
f27	27	17	7	12	91	19	73
f39	39	N/A	76803	31903	N/A	N/A	323599
39a	39	24	27	16	127	169	73
39b	39	N/A	16935	4946	N/A	143098	29068
39c	39	28	20	9	131	115	13
39d	39	44	29	24	195	208	109
39e	39	9490	5034	741	46847	46789	3718
42a	42	14359	7093	2348	65345	52468	12310
42b	42	51916	17670	9847	120969	142495	59497
45a	45	19	9	11	59	13	25
45b	45	5720	5226	5573	22121	33475	33484
51a	51	558	58	105	2169	259	439
51b	51	6832	2962	554	29471	17557	2296
eil51	51	4216	1010	284	15243	4984	1141
f99	99	N/A	3667	4055	N/A	10546	7228
rat99	99	N/A	9692	3444	N/A	23731	7012

Table 2: A comparison of the branching rules

corresponding vertex is included in K_0 and all the elements of the row are fixed to 0.

4. If there is only one free variable in a column, then it is fixed to 1. The necessary additional operations are also performed, i.e., the index corresponding to the row of the fixed variable is included in K_1 if it has not yet been included.

Finally, some feasibility tests are also performed after each subgradient iteration, i.e., if too many medians or non-medians have been fixed by the deterministic rules, or all the elements of a column have been fixed to 0, or there is no free point to allocate to a previously identified median, then the corresponding tree node will be fathomed.

Now we have the components for a complete branch-and-bound procedure. In the light of the previous experiments on the quality of the lower bounds, we will apply the bound which is based on our third relaxation (LR3). Next we examine the efficiency of the branching rules, see Table 2. All three procedures apply the penalties and the additional variable tests when calculating the lower bound. They differ only in the branching rules they apply, and we denote them by "P1", "P2" and "P3", respectively. The running times[2] (in seconds) and the number of examined nodes are indicated in Table 2 for the selected problem instances.

The results show that the second and third branching rule have significantly better performance than the first rule. Furthermore, the third rule is better than

[2]All computational tests were performed on a 133MHz Pentium PC.

		Time (sec)			Nodes		
Problem	Pts	P3	P3-	P3- -	P3	P3-	P3- -
f21	21	1	2	19	1	16	40
f27	27	12	26	187	73	244	436
f33	33	1	1	3	1	1	1
39a	39	16	28	298	73	169	346
39c	39	9	29	171	13	181	139
39d	39	24	32	275	109	160	268
39e	39	741	1617	26072	3718	10867	17242
45a	45	11	15	205	25	43	130
51a	51	105	675	7515	439	4096	5284
51b	51	554	4086	32615	2296	22360	23635
eil51	51	284	851	11843	1141	4663	9352

Table 3: The effect of the penalty tests and the additional variable tests

the second one, and therefore, the application of that rule is desirable in an efficient branch-and-bound procedure. The average ratio of the running times of the procedures P1 and P3 was roughly five; for P2 and P3 the corresponding value was two. On average, the procedure P1 generated ca. five times more nodes than P3, while the average ratio of the number of nodes for P2 and P3 was about three. The average processing time per a node was approximately the same for all three procedures.

Concerning the efficiency of the penalties and the additional variable tests, Table 3 presents a comparison which is based on some of the easier problem instances. In the light of the experiments with the branching rules (see Table 2), the third branching rule is applied in all three procedures. The procedures differ only in the way they apply the penalties and the variable tests: "P3" denotes the variant which applies both the penalties and the additional tests (this variant was also included in the previous tests), "P3-" denotes the variant where the penalties are applied but the additional variable tests are not, "P3- -" denotes the variant without the penalties and without the additional variable tests. The table indicates the running times in seconds and the number of examined nodes for the selected problems.

The results of Table 3 clearly show the advantages of applying both the penalties and the additional variable tests. The penalties generally enable to fix a large portion of the variables and have two main advantages. Firstly, by fixing many of the variables, the calculation of the lower bound becomes much faster. Secondly, the performance of the branch-and-bound procedure based on the lower bound equipped with the penalty tests is much better, since eliminating many of the free variables reduces the size of the enumeration tree.

The average ratio of the running times of the procedure without the penalties (P3- -) and with the penalties (P3-) was roughly ten, while the average ratio of the number of nodes was two for the same procedures. Concerning the effect of the additional variable tests, the average ratio of the running times of the procedure

without these tests (P3-) and with these tests (P3) was roughly three, whereas the average ratio of the number of nodes was six for the same procedures. The average processing time of one tree node was approximately 3.5 seconds for P3, 5.7 for P3-, and 1.0 for P3- -. These values correspond to the observation that the application of the penalties demands more processing time. The number of nodes is, however, much smaller when penalties are applied. The additional variable tests speed up the calculation, and their application also yields a smaller number of tree nodes on average. To summarize, the experiments confirm that the application of both the penalties and the additional variable tests is advantageous.

6 Comparisons to other approaches

In this section, we compare the performance of P3 with our previous B&B variant ("P-LB1") [6] and with general MIP solver packages "OSLMIP"[3] and "lp-solve"[4]. The P-LB1 is based on the quickly computable lower bound LB1. The branching is done by selecting the free variable with the smallest cost and fixing its value either to 0 or to 1 in the two descendant nodes. The P-LB1 also performs some variable manipulating procedures, which can fix certain additional variables on the basis of the previous branching decisions.

Table 4 shows the running times for the different procedures in seconds. The better solver (OSLMIP) was performed on all the three integer programs, i.e., on IP1, IP2a and IP2b, while the other (lp-solve) is performed on IP1 only.

The results of lp-solve with IP2a and IP2b are not included in Table 4, because they were much worse than with the formulation IP1. We could obtain results only for the first two problems. These results were roughly 6-10 times worse than the ones with IP1. For the other problems, we interrupted the execution of the procedures after 8-10 hours without termination. These observations are especially interesting if we compare them with the results of the other solver, which showed that the latter two formulations are much more promising to solve.

The results of Table 4 show that the procedure P3 outperforms the procedure P-LB1 due to the sharper bounding function. The calculation of the specific bound of P-LB1 is much faster than the bound of P3, but the improvement in the quality of the bound yields a much smaller tree size and total running time.

Concerning the comparison with the general MIP solvers, the results are diverse. The highly optimized machine code of the first solver (OSLMIP) gives better results than our procedure, while the general package (lp-solve) was much worse than our method. We think that differences in the running time between our procedure and the better solver may lay mainly in technical details, as we used general techniques and high-level programming languages for the implementation. The ratios between the running times of P3 and OSLMIP show that the difference does not grow with

[3]The package called OSLMIP is developed by IBM and it is one of the market leaders with respect to the performance. We downloaded the 60-day trial version from http://ism.boulder.ibm.com/es/oslv2/startme.htm

[4]The package lp-solve can be downloaded from ftp://ftp.ics.ele.tue.nl/pub/lp_solve/

Problem	Pts	OSL-IP1	OSL-IP2a	OSL-IP2b	P-LB1	P3	LP-IP1
f21	21	10	7	4	8	1	34
f27	27	19	8	12	59	12	386
f33	33	10	4	4	98	1	181
f39	39	8114	620	814	122689	31903	N/A
39a	39	76	16	43	67	16	98
39b	39	7642	644	206	74035	4946	N/A
39c	39	71	13	15	2541	9	21452
39d	39	32	14	27	1000	24	2369
39e	39	1220	402	160	59605	741	20435
42a	42	3670	584	406	31272	2348	231900
42b	42	5787	452	543	N/A	9847	N/A
45a	45	88	16	25	N/A	11	N/A
45b	45	423	107	62	N/A	5573	N/A
48a	48	9	9	7	145	1	200
48b	48	39	8	8	N/A	3	N/A
51a	51	130	23	31	186560	105	N/A
51b	51	841	302	88	N/A	554	N/A
eil51	51	777	613	133	N/A	284	N/A
f99	99	2889	718	1312	N/A	4055	N/A
rat99	99	N/A	3545	2100	N/A	3444	N/A

Table 4: A comparison of some methods for finding the exact solution of the 3MP

the problem size but stays at a constant level. However, for some hard problems this difference can increase, see problems f39, 39b, 42b, 45b.

Experiments with problem instances with different cost matrix show a rather different trend [11]. It turned out that matrices with the triangle inequality property are hard for our procedure, while on totally random cost matrices our procedure is superior to the other approaches.

Comparing the efficiency of the MIP solvers with respect to the applied integer program, as stated above, it is important which formulation is used as an input. The results of OSLMIP show that the modified p-median formulation is much better. Moreover, IP2b seems to be slightly better than IP2a.

7 Conclusions

We have considered the exact solution of an NP-complete graph problem, the three-matching problem. We introduced a problem-specific lower bound and a Lagrangian lower bound based on an integer formulation to the problem. Based on the connection to the p-median problem [2], two further Lagrangian relaxations were given to the problem. We gave a solution of the 3MP by modifying an algorithm for the p-median problem.

Our empirical results showed that the lower bounds were quite different, and special care must be taken when selecting the constraints to be relaxed to the

objective function. It was also shown that two seemingly different approaches gave the same lower bounds (LR1 and LR2). The application of the penalty tests speeds up the calculation of the lower bound considerably because it eliminates a large portion of the free variables.

We constructed several complete B&B procedures by introducing different branching rules. Our modification of the branching rule of [2] gave the best performance. Furthermore, we incorporated additional variable tests in order to enable to fix more free variables. The efficiency of the branching rules were examined and the application of the penalties and the additional variable tests were justified by empirical tests.

We compared our B&B procedure with our previous algorithm, which uses a quickly computable but worse lower bound, and the results clearly showed that the increase in the quality of the bound was essential and the procedure with the better bound outperformed the previous approach. Furthermore, we compared our procedure with two general optimization packages, and the results were diverse. The highly optimized machine code of a commercial MIP solver had better running times than our procedure, but the ratios of the two running times seemed to remain constant. The other solver, which was written in C, had a worse performance than our procedure.

It was also shown that it was very important which integer program was used in the MIP solvers. There is a significant difference between the formulations IP1 and IP2, but even the small difference between the formulations of IP2a and IP2b can yield great differences in the performance of the solvers applied to the two programs.

References

[1] E. Balas and M.J. Saltzman: An algorithm for the three-index assignment problem, *Operations Research 29 (1991)*, 150-161.

[2] N. Christofides and J.E. Beasley: A tree search algorithm for the p-median problem, *European Journal of Operational Research* 10 (1982), p. 196-204.

[3] Y. Crama, A.W.J. Kolen, A.G. Oerlemans and F.C.R. Spieksma: *Production Planning in Automated Manufacturing*, Springer-Verlag, 1994.

[4] M.L. Fisher: The Lagrangian relaxation method for solving integer programming problems, *Man. Sci. 27 (1981)*, 1-18.

[5] M. Johnsson, T. Leipälä: Determining the manual setting order of components on PC-boards, *Journal of Manufacturing Systems*, Vol. 15 No. 3 (1996), 155-163.

[6] M. Johnsson, G. Magyar, O. Nevalainen: On the Euclidean 3-Matching Problem, *Nordic Journal of Computing* 5(1998), p. 143-171.

[7] P. J. M. van Laarhoven and W. H. m. Zijm: Production Preparation and Numerical Control in PCB Assembly, *The International Journal of Flexible Manufacturing Systems*, 5 (1993), p. 187-207.

[8] E.L. Lawler and D.E. Wood, "Branch-and-Bound Methods: A Survey", *Operations Research 14 (1966)*, 699-719.

[9] G. Magyar, M.Johnsson, O. Nevalainen: Genetic algorithm approach for the three-matching problem, *Proceedings of the Third Nordic Workshop on Genetic Algorithms and their Applications (3NWGA)*, p. 109-122, Aug. 1997. (ftp://ftp.uwasa.fi/cs/3NWGA/Magyar.ps.Z)

[10] G. Magyar, M.Johnsson, O. Nevalainen: An adaptive hybrid GA for the 3-matching problem, *Turku Centre for Computer Science (TUCS) Technical Report 166*, March, 1998.

[11] G. Magyar, M.Johnsson, O. Nevalainen: On the exact solution of the three-matching problem, *Turku Centre for Computer Science (TUCS) Technical Report 199*, October, 1998.

[12] K. Palletvuori, P. Luostarinen, K. Muurinen and O. Nevalainen: "On the scheduling of a multipurpose laboratory analysis instrument", In *9th Euromicro Workshop on Realtime Systems 97*, 1997.

[13] C.H. Papadimitrou and K. Steiglitz: *Combinatorial Optimization, Algorithms and Complexity.* Prentice-Hall, New Jersey, 1982.

[14] *Modern Heuristic Techniques for Combinatorial Problems*, edited by C.R. Reeves. McGraw-Hill, 1995.

[15] G. Reinelt: TSPLIB - A Traveling Salesman Problem Library, *ORSA Journal on Computing 3 (1991)*, 376-384.

Acta Cybernetica 14 (1999) 377-387.

Testing Internet Applications - Terminology and Applicability

Mazen Malek * Roland Gecse *

Abstract

This paper examines the applicability of OSI conformance test methodology to Internet protocols. It summarizes the differences between them and introduces the Internet Reference Model along with a new abstract test method, which was designed for the practical purposes of conformance testing of TCP/IP protocols. Some interesting test cases and points, that were chosen from RIP, demonstrate the facilities of the model and give impression of testing Internet protocols.

1 Introduction

Up to now, in the Internet community, conformance testing was an unknown concept. However, the need for recommendation conforming TCP/IP implementations grows, as the application of Internet protocols in business telecommunication systems is becoming reality. It is probable that more and more vendors are going to provide Internet products, whose reliability and interoperability with other products have to be assured.

Although conformance testing methodology [1] was originally intended for OSI based systems, there are ongoing discussions about its applicability to the TCP/IP protocol stack. Numerous articles and conference contributions justify that these questions present a current topic. [2] founds theoretical base of relay system testing, which is then used, among others, for the testing of Simple Mail Transfer Protocol [3] and IP router. [4] and [5] focus on detailed analysis of Transmission Control Protocol's flow control algorithms that are expected to be used in measuring and fixing the majority of implementation problems listed in [6]. On the other hand, [7] deals with interoperability test suite derivation that may be used for the purpose of Internet testing.

The following issues, beside others, will be argued in this paper. Sections 2-5 give an overview of the Internet protocol structure, introduce the Internet Reference Model and suggest a new abstract test method. Also, similarities and differences in layering, data flow and configuration are fetched in comparison to the OSI Basic Reference Model (BRM) [8]. After the presentation of a possible test realization

*Conformance Center, Ericsson Ltd., Laborc u. 1., H-1037 Budapest, Hungary

(section 6) and a short overview of the Routing Information Protocol and related Internet routing concepts (section7), sections 8 and 9 give some practical testing experience.

2 Comparing Internet and OSI architecture

The OSI BRM has 7 layers, each of which with a well-defined task. OSI protocol stacks are designed to fit to this model. The protocol entities (PEs) of a particular protocol suite are associated to the appropriate layers. Peer-to-peer communication between two PEs of the same layer takes place in abstract protocol data units (PDUs) while physical communication with upper and lower layers' PEs is only possible via service primitives (SPs).

Unfortunately, Internet was not planned to have such a detailed abstract model. The structure of TCP/IP, which represents the actual state of Internet, has evolved gradually from the beginning [9]. Internet has only four layers: link, network, transport and application. Although the general functions of these layers are not as well defined as OSI's, they provide almost the same functionality. Disregarding that reliable service appears first only in the transport layer, network and transport layers map to their OSI counterparts. Internet link layer maps, in general, to OSI physical and data-link layers. Since the application layer holds all remaining functionality (OSI layers 5-7), applications may gain enormous complexity. Internet protocols do not have standardized SPs, thus in contrast to open systems; the communication between neighboring layers is implementation specific. This, besides the loosely specified layer characteristics, results that layer boundaries are flexible. Another feature that must be kept in mind when talking about Internet is the whole TCP/IP protocol stack should be considered as a single unit together with a set of alternative protocols. The transport layer for example consists of two protocols: the transmission control protocol (TCP), which is a connection-mode service and the user datagram protocol (UDP) that provides a connectionless service. In a particular communication process, at most one of these services is used.

From the configuration point of view a real open system can act as end system, relay system or both simultaneously. Internet systems have also this kind of configurations with noting that relay systems are called also intermediaries. Intermediaries are further subdivided according to working aspects to proxy, gateway and tunnel. In our paper we limit our discussion to relay systems and call them routers.

3 Conformance testing of Internet protocols

From the conformance testing perspective it is worth to distinguish between hardware and software implementations. Hardware implementations (e.g. IP router) neither implement the whole TCP/IP protocol stack, nor provide interface to protocol layers. Accordingly, they could be examined only by an external test system. Software implementations (e.g. FTP client, httpd programs), on the other

hand, have numerous advantages over hardware systems. Besides the existing test methods [12], they imply the possibility of designing more effective new test methods. For the understanding of these methods, a particular TCP/IP implementation should be examined.

4.4BSD-Lite's Net/3 networking code [10] can be considered as a reference implementation of the Internet protocol suite (Besides TCP/IP, it also supports Xerox Network Systems (XNS), OSI communication protocol families and the Unix domain protocols that are provided for Inter Process Communication (IPC)).

The structure of the Net/3 networking code is presented in figure 1. Application level protocols (FTP, Telnet, and RIP) are distinguished from the underlying TCP/IP stack. They are running as processes in the device's user space while underlying layer protocols used to be implemented as a single unit in the operating system space. The internal structure of this unit consists of three layers: application programming interface (API) or socket layer, protocol layer and interface layer. The public functions of this unit can be reached at the kernel entry points using system calls (SCs) which represent the operating systems' service primitives. API, in addition to separating the application layer, provides a protocol independent interface to the entities of the underlying protocol layer. It offers a set of different networking features of the kernel that can be reached uniformly via SCs. The protocol layer holds the Internet transport (UDP, TCP) and network (IP, ICMP, IGMP) layer protocols [11]. The protocol layer does not provide SCs to application layer entities. The interface layer consists of various device drivers implementing link layer protocols (e.g. Ethernet) and procedures that are used for address conversion between the protocol layer and it. The code for different pseudo devices (loopback interface, BSD packet filter (BPF)) can also be found there. Interface layer functions are accessible through SCs. The packet filtering functions are further applicable for control and observation. Now, having a global picture of the overall structure of TCP/IP, the Internet Reference Model will be introduced.

4 The Internet Reference Model

It can be stated that all of today's software TCP/IP implementations are based upon the architecture of Net/3. By considering this, a model will be introduced that is suitable for conformance testing and incorporates the listed features of software implementations. In the Internet Reference Model (IRM), In Figure 1, the functions of SPs are replaced by SCs of API (In this context, API is used as a general term, which in a particular implementation (e.g. Net/3) stands for both socket API and BPF. That is, because the socket API does not provide access to the interface layer). These SCs allow applications to send PDUs directly to each layer protocol entity. The API itself should be considered as a switch that connects applications to the selected underlying service via SCs. The functions of the API are provided at kernel entry points (rhombus). The semicircles present the possible destination protocol layers to which SCs provide access. The dashed line expresses that API itself is not a protocol. Although the IRM has some minor differences

from OSI BRM, which are coming from design aspects, the applicability of existing conformance test methodology is straightforward.

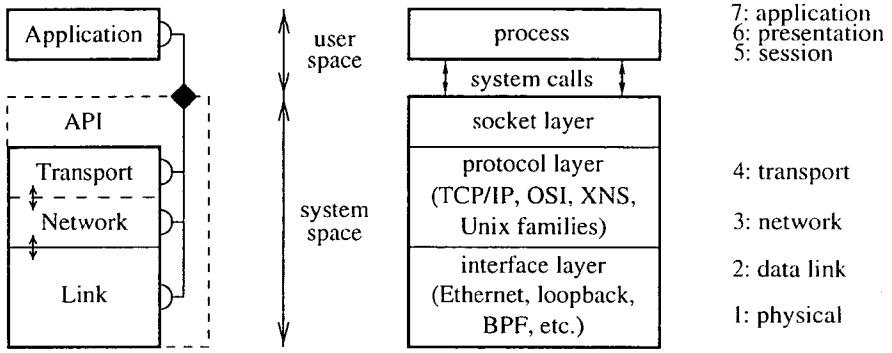


Figure 1: The Internet Reference Model (left), general organization of Net/3 networking code (right)

5 Abstract Test Methods

Considering the open structure of software implementations, the new Joint Test method (JT) will be defined, which can be uniformly applied to testing of all protocols of IRM. JT can be applied both in Single Party Testing (SPyT) and in Multi Party Testing (MPyT) context. When used in SPyT, it resembles to the local [1] test method, whereas the MPyT variant has similarities to the local transverse test method in [2].

JT is shown in (Figure 2), and uses the graphical notation of [12].

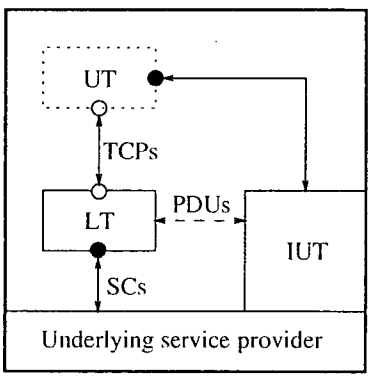


Figure 2: The joint test method

JT has the following characteristics:

- Test system and system under test (SUT) are on the same system.
- There is an optional Upper Tester (UT), and one Lower Tester (LT) in SPyT; no UT, an arbitrary number (usually 2) of LTs and a Lower Tester Control Function (LTCF) in MPyT. UT, LT(s) and LTCF are application layer processes.
- The Points of Control and Observation (PCOs) are at the LT and UT.
- Test coordination is done using Unix IPC.
- Test events are exchanged in PDUs using SCs of API. The control and observation is provided by means of API.

The most significant difference to the ancestor test methods, which is very advantageous in practical testing of software TCP/IP implementations, is that LT(s), UT and coordination procedures are placed in the application layer regardless of the layer which is occupied by IUT. Another good feature is that JT can be applied to both end systems (SPyT) and relay systems (MPyT), thus intermediaries can be tested out of their environment.

6 Test realization

Having an implementation to be tested and an abstract test suite (ATS), the means of testing should be provided. It consists of the implementation of tester functionality, the derivation of ATS into executable test suite (ETS) and the production of test documents. System Certification System (SCS) is a set of tools provided by Ericsson that can be used in a wide variety of testing: functional testing (white-box technique), conformance and interoperability testing (black-box) and performance testing (white/black-box). SCS is based on the following principles:

- Protocol independence. This means that different protocols can be tested on the same manner.
- Multiple simultaneous protocols. Not only one but also many protocols can be accessed from the same test.
- Distribution. One test may be distributed (over the Internet), making it possible for each part of the test most closely related to one interface to reside in the box containing that physical interface.
- Platform independence. SCS is independent of the platform in which the SUT executes. It can execute the same tests both against the physically real SUT and the SUT only simulated in a workstation (bypassing the lowest protocol layers).

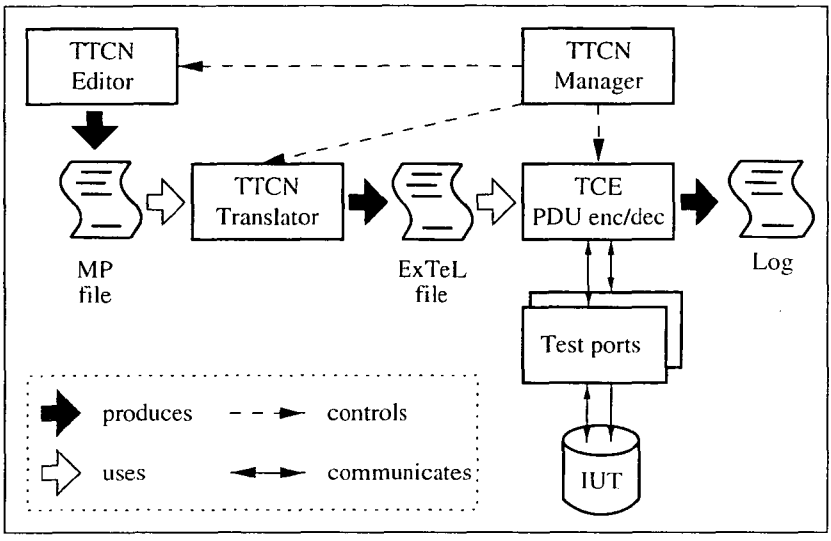


Figure 3: SCS structure

One of the main ideas in SCS is that it is an interpreting execution platform. This means that a TTCN test suite (an MP file) given as input to the Translator is first converted into an intermediate language, ExTeL (Executable Test Language), which then can be directly executed (interpreted) by the ExTeL Test Component Executor, TCE (see also Figure 3 above). With this method there is only one phase from a TTCN test suite to the final executable format which makes it different compared to the compiling methods, where an extra compilation and linking phase has to be performed. Another important feature in SCS is the Test Port concept. With this solution it is possible to develop the core functionality separately without affecting the existing test ports and vice versa. For this reason, a complete set of test ports where realized and worked out by the authors, particularly, IP, TCP and UDP test ports. There exist also two built-in PDU encoder/decoders: BER (Basic Encoding Rules) and a raw binary encoder/decoder. TTCN Manager is the front end in SCS. It has the control over execution and monitoring. The log files for different test components can be observed in real time.

7 Routing techniques

Routing involves two basic activities: determination of routing paths and the transport of information groups (packets) through an internetwork. In some literatures the later is referred to as switching. Path determination, on the other hand, can be very complex. To aid the process of path determination, routing algorithms initialize and maintain routing tables, which contain route information. Usually,

routing algorithms fill routing tables with destination/next hop associations. These associations tell a router that a particular "destination" can be gained optimally by sending the packet to the node identified in "next hop". Routers communicate with one another (and maintain their routing tables) through the transmission of a variety of messages. The routing update message is one such message. Routing updates generally consist of all or a portion of a routing table. They are the means by which routers communicate path information between one another. In this paper we limit our illustration on a very common routing technique, it is the RIP protocol. It is a distance vector, intra-domain routing protocol originally designed for PUP (Xerox PARC Universal Protocol, 1980) and used in XNS. RIP became associated with both UNIX and TCP/IP in 1982 when the Berkely Standard Distribution (BSD) implementation of UNIX was introduced.

8 Interesting issues in testing routing

When time comes and routing testing is to be performed, a lot of interesting points should be considered. As the basic representation of test cases is the stimuli-response pairs, one should define precisely the types and instances of such pairs. Accordingly, routing stimuli and responses were defined. We have found that we need to deal with two different layer concepts. Firstly, the Internet layer and its datagrams that is used to check the arrival of forwarded data. Secondly, the application layer, or the routing application in question. Within this context, majority of testing events will be written by applying types and instances of this application. Both stimuli and responses, at application level, are in a form of routing updates. These updates can be gotten at prescribed time intervals or synchronized to other testing events. Another point of interest is the alternatives to those responses. They may have the form of modified routing update, in terms of metrics and/or routes.

A conformance test case has, typically, three phases of actions, preamble, test case body and postamble. The first and third phases are for state transition, to initial testing state and to idle state respectively. In the case of routing application, state inquiry is only possible when the routing table is to be accessed, through the usual updates or by asking the table on demand.

In the Internet, routing is the main building block in its backbone. That's why they play a crucial role when we think of the enormous growth of the Internet. Another important issue here is that routing techniques are evolving more quickly than host ones.

9 Example

In this section we try to illustrate our method of testing on a simple configuration of three test components connected to the IUT. Each test component has the role of simulating another router in the network, while connections through the configuration tries to indicate the subnetworks involved in the testing. According to

the standard [13]: "An internet router should be capable of choosing a next-hop destination for each IP datagram", and according to standard [14]: "After the validation process of a response finishes successfully, an update that changes the metric of an existing entry in the routing table should be a trigger for entry modification". This modification varies between recalculating the metric according to the following formula:

$$metric = MIN(metric + cost, 16)$$

where 16 indicates that a link is inaccessable, and adding a new entry. In our example, we have established the tested condition as if one route is substituted by the other. In particular, we have informed the IUT that the desired destination (address included in an IP datagram) is reachable with a lower cost through B router instead of C router. In terms of Tree and Tabular Combined Notation (TTCN) [15], the official test notation of the conformance testing, a preamble will contain the preliminary parts of testing; i.e. getting the configuration working, sending an update form B router (within the testing context it is a test component) with a cost of the destination less than what is sent by C, and sending an IP datagram from A with the destination address. Test case body, however, is a set of test events that is responsible for issuing a test verdict, indicating the correctness of behaviour. Here, it is the arrival of the sent IP datagram at B instead of C. Before ending the test case, an action should take place to get the implementation right to its original state. This is possible by refreshing the routing table with the original route update, in terms of RIP it means a new response from C with higher cost or form B with lower cost. Figure 4 demonstrates the overall distribution of testing functionality. In the laboratory environment, testing was handled with software components, called gatewas, and with the standard SCS interface together with the newly defined test ports. Accordingly, IP test port was used to transfer the sample IP datagram and UDP test port to send and receive routing updates. The two solutions differ in the appearance of test cases. In the first solution, test cases where included in the code part of the gateways, while the other used an interface to a TTCN editor. In short we list below the tasks associated to test components.

Tasks of test components:

- Simulate RIP
- Test coordination & management (remote/local)
- Execute Test Cases

According to this example, the following Test Purpose was defined.

Test Purpose 1.1:
"To check if the router applies to changes in network topology after receiving the required update from other routers"

Abbriviations:

- IUT - Implementation Under Test
- A, B, C - gateways
- a, b, c, ai, bi, ci - networks

Roles of gateways

- Client
- server
- Listener

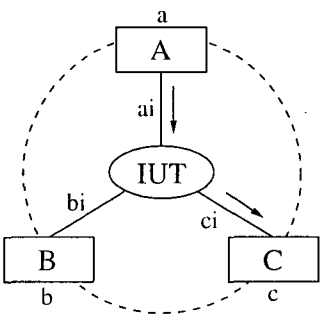


Figure 4: Illustration of a test application for IP routing

10 Conclusions

In this paper, differences between OSI and Internet systems were summarized. Then the Internet Reference Model was introduced together with the Joint Test method for conformance testing (which is major contribution to the base methodology). Afterwards, a practical application of the new concept was demonstrated on the testing of RIP routing protocol, which is a continuation of the progress reached in [16]. The Test Purpose viewed here is part of a complete set to provide complete testing for routing applications. We have shown a framework for testing Internet protocols, which was worked out on the basis of the conformance testing framework of [1]. Experiences with testing RIP showed that some extensions are necessary to TTCN for making it more suitable to describe test cases for testing Internet protocols. This is true especially for testing performance related features of the product. Our interest for the time being is to generate a complete test suite to test a routing application, RIP2 or OSPF. Future work can be for example the interoperability testing based on this concept of Internet protocols, and introduction of formal extensions that are more suitable for Internet testing.

References

[1] ITU-T X.290–X.296, OSI conformance testing methodology and framework for protocol recommendations for ITU-T applications, 1994-1995.

[2] Bi, J. and Wu, J.: Towards abstract test methods for relay system testing. Testing of Communicating Systems, Volume 10 pp 381-397, IFIP, 1997.

[3] Bi, J. and Wu, J.: Application of a TTCN based conformance test environment on the Internet email protocol. Testing of Communicating Systems, Volume 10 pp 324-330, IFIP, 1997.

[4] Kato, T., Ogishi, T., Idoue, A. And Suzuki, K.: Design of Protocol Monitor Emulating Behaviors of TCP/IP Protocols. Testing of Communicating Systems, Volume 10 pp 416-431, IFIP, 1997.

[5] Kato, T., Ogishi, T., Idoue, A. and Suzuki, K.: Intelligent Protocol Analyzer with TCP Behavior Emulation for Interoperability Testing of TCP/IP Protocols. Formal Description Techniques and Protocol Specification, Testing and Verification, FORTE X/PSTV XVII '97 pp 449-464, IFIP, 1997.

[6] Paxton, V. (editor), Allman, M., Dawson, S., Heavens, I. and Volz, B.: Known TCP Implementation Problems, ¡draft-ietf-tcpimpl-prob-02.txt¿, Internet Draft, IETF Network Working Group, May 1998.

[7] Malek, M. and Dibuz S.: A Pragmatic Method for Interoperability Test Suite Derivation. EUROMICRO'98, Proceedings of the 24th Euromicro Conference, Stockholm, Sweden, 1998. Aug 24-26.

[8] ITU-T X.200, Information Technology - Open Systems Interconnection Basic Reference Model: The Basic Model, 1994.

[9] Carpenter, B. (editor): Architectural Principles of the Internet, RFC 1958 Informational, IETF Network Working Group, 1996.

[10] Wright, G. R. and Stevens, W. R.: TCP/IP Illustrated, Volume 2, The Implementation. Addison-Wesley, 1995.

[11] Stevens, W. R. TCP/IP Illustrated Volume 1, The Protocols. Addison-Wesley, 1994.

[12] Baumgarten, B. and Giessler, A.: OSI conformance testing methodology and TTCN, North. Holland, 1994.

[13] P. Almquist:. Towards requirements for IP routers, Network working group, RFC1716, November 1994.

[14] C. Hedrick: Routing Information Protocol, Network working protocol, RFC1058 June 1988.

[15] ISO/IEC 9646-3, The Tree and Tabular Combined Notation (TTCN), 1998.

[16] Gecse Roland: Conformance testing methodology of Internet protocols, Internet application-layer, Protocol testing- the Hypertext Transfer Protocol. The IFIP 11th International workshop on Testing of Communicating Systems IWTCS'98, August 31- September 2, 1998, Tomsk, Russia

Acta Cybernetica 14 (1999) 389-402.

A study of portability in the deployment of WWW

András Micsik *

Abstract

There are several problems with handling compound hypermedia documents on the Internet currently. One class of these problems, namely portability is studied in this paper in detail, and possible solutions are examined. The offered solutions are based on a new container architecture, the WebPack format, currently under development at SZTAKI.

Keywords: hypermedia documents, document-like objects (DLO), World Wide Web, portability, metadata, Internet

1 Introduction

As the World Wide Web [1] spread the world from the beginning of this decade, it incorporated more and more powerful tools and formats, and the information served via WWW became more and more complex. The content and layout of WWW pages became competitive with printed material, and in other aspects WWW pages have far more potential than printed documents. Searching, interactivity, animations and virtual reality are just keywords to make the additional possibilities felt.

The meaning of document in case of the Internet is changing. Digital documents are sometimes more similar to a piece of software than to printed material. Furthermore these documents on the Internet are interconnected with each other via hyperlinks. The Dublin Metadata Workshop [15] investigated this new kind of information source, and created a new term: Document-like Object (DLO) [14]. A DLO can be characterized like this:

- it may contain files in lots of different formats: text, graphics, animation, video, audio and 3D models
- its files and data are interconnected with hyperlinks.
- it may contain executable parts (applets, scripts, objects, etc.)

*Department of Distributed Systems, MTA SZTAKI, 1111 Budapest XI. Lágymányosi u. 11. Hungary, e-mail: micsik@sztaki.hu

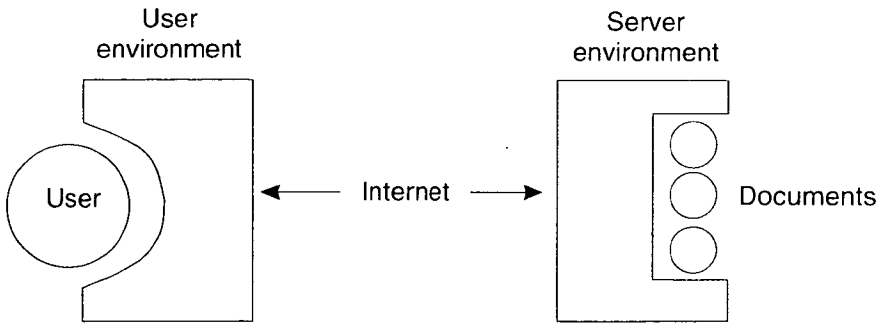


Figure 1: Using WWW documents

DLOs are called WWW documents or simply documents for simplicity in this paper. WWW documents can be surprisingly immovable compared to other widely used formats such as Postscript, Microsoft Word or ToolBook. This is because these documents may depend on the presence of several files and programs, and they may also depend on services of the WWW server. Currently there is no general approach to provide collection and management of these dependencies and therefore the management of WWW documents is solved individually by web administrators. Management in general is not in the scope of this paper, rather one of its subtopics, portability. When moving a document from one host to another, all dependencies in a WWW document must be explored and fixed in order to restore correct operation. The nature of portability dependencies is analyzed in Section 2. A formal descriptive framework for poratbility conditions is given in Section 3, and technical solutions for the identified problems are given in Section 4. An outlook to related work and the summary concludes the paper.

2 Analysis of portability issues

Users access WWW documents through the user and server environments. The *user environment* is specific to the actual user of the document, while the *server environment* is specific to the used document. The user environment provides browsing and viewing capability for the user. The server environment offers the services of the operating system and the WWW server for the document. A server environment usually serves several documents.

The *correct operation* of a WWW document means that all its features (hyperlinks, images, interactive parts, etc.) are available for the user in the same way as the author has implemented them. This relies on both the user and server environments, and may rely on other WWW documents or external data as well. As the user environment is totally under the control of the user, this approach concentrates on the correct operation of the server side. If the operation of the server side is based on open standards, it is possible to give recommendations for the user

environment as well.

In case of a WWW document *portability* means that if the document is placed into a new server environment, or its server environment has changed, there is a way to keep the same operation of the document as in its previous environment. In the next subsection the most generally used formats and technologies are listed together with the problems that jeopardize the portability of WWW documents.

2.1 Summary of formats, technologies and portability problems

URIs and URLs Embedded or linked objects are most commonly referred to as Uniform Resource Locators (URLs) [5] on the Internet. A URL can refer to

- an object on the same host
- an object on a different host
- a part of an HTML page
- dynamically created objects

The problem with URLs is that they are location specific. The referred object is identified by the combination of the host machine name and the descriptor of the object in the file system. This means that moving a part of a document to another location in the file system or to another host can make that part inaccessible.

URLs are a subclass of Uniform Resource Identifiers (URIs) [2]. The URI schema defines a highly customizable reference methodology for the Internet, which would allow location transparent naming facilities, but currently there isn't any widely used and location transparent naming facility for the Internet.

HTML The Hypertext Markup Language [3] serves as a basis for WWW documents. It provides embedding and linking facilities for other digital formats, and usually acts as the main user interface for the whole document. After moving an HTML page to another location URLs for inline images, or URLs to other pages may become incorrect making the page unusable and the referred pages inaccessible.

CGI The Common Gateway Interface [6] is a simple and general mechanism to call executable programs from static WWW pages. The program is identified as a URL, and there are two ways to pass parameters to the program: encoded in the URL or via HTML forms. When the program is launched the calling parameters are passed, and the program output is an object to display for the user. CGI scripts can work only with the help of a WWW server, but the scripts are executed in the context of the operating system. Therefore any kind of compiled or interpreted program can be used to write CGI scripts. For example Perl is a very popular language of CGI.

CGI scripts mean a great problem concerning portability, because the execution is controlled only by the operating system. The program may rely upon other programs or data files, and these may not be present on another host.

Applets Applets are written in Java language [7], and provide a full-featured graphical user interface in a part of the screen during WWW browsing. Java is an interpreted and platform-independent language, so an applet should execute in the same way in every WWW browser on every machine (in practice there are small, not very significant differences among different browsers and different operating systems).

Applets can load parts of their code and additional data files from the server host, and can communicate directly with other network services on the server host. Therefore hidden file dependencies can still occur similarly to CGI scripts.

JavaScript JavaScript [8] offers a way to associate custom code with HTML pages, and in this way to add new functionality to WWW browsers. For example with JavaScript the programmer can change the behaviour of a button in an HTML form, or can load other pages automatically into the browser. Unfortunately JavaScript is not one uniform scripting language. It has no specification, and its support differs remarkably in different browsers (namely Netscape and Internet Explorer) and even in different browser versions.

Server side includes There are several very different tools that are described here commonly as server side includes. The basic server side include facility is a way to include parts into a HTML page on the fly when the page is accessed by the user. Traditionally there were two possibilities: to include another file or to include the output of a program. Later these possibilities were enhanced to enable flexible scripting and database access. Products falling into this category are for example PHP and Microsoft's Active Server Pages. From the portability viewpoint server side scripts require the presence of their specific interpreter environment, and may introduce additional dependencies on data files and executables.

VRML With the use of the Virtual Reality Modeling Language [9] one can integrate 3D models into a WWW document. These 3D models may contain animations, interactive scripts, and hyperlinks to other objects. Models often incorporate external objects (images, sounds, etc.). Links and embedded objects are both represented as URLs, so in this aspect VRML holds the same problems as HTML, furthermore VRML can contain scripts as well, inheriting the portability problems with executables. VRML files are sometimes stored in a compressed format to speed up download. To find external references, these files must be uncompressed temporarily.

The server environment The server environment provides elementary document services called features later in this paper. These include authentication and

access control, handling of object types, aliasing or redirection of URLs, and server side includes or scripting. The configuration of these features differs from server to server, and often needs the editing of complicated configuration files. Additionally the server environment provides a set of installed auxiliary programs: scripting tools, image manipulating software, etc.

The user environment This environment is not only the WWW browser but also a set of external viewer applications for various formats (e.g. Postscript, PDF or VRML) that cannot be viewed within the browser. Also the browser preferences set by the user are very important, here viewers are associated with formats, Java or JavaScript is enabled or disabled, etc. Finally the type and version of the WWW browser determines the capability of JavaScript and Java execution.

3 Describing portability conditions

The first step towards the solution of the above mentioned problems is the formal description of portability conditions. The description is based on the dependence and reference relationships. With these relationships all aspects that are important for portability can be formalized. For these relationships a metadata-based representation is given for automatic processing, and a graphical representation for humans. Metadata literally means data about data [15], data characterizing an object. Typically metadata provides descriptive, administrative and structural information about an object.

The descriptive framework for portability conditions is built with the following terms: the server environment contains a set of WWW documents. A *WWW document* is defined as a set of objects. *Objects* are single-file entities in any format. Each entity (objects, documents, etc.) may have metadata associated with it. *Metadata* describes either relations among objects or properties of objects. Relations may have properties as well. *Properties* are simply name-value pairs.

3.1 Definition of dependence and reference relationships

Saying object X *depends on* object Y means that the correct access and operation of object X requires that object Y is accessible and operates correctly for object X. This definition of "depends" is the transitive closure of the dependence relationship.

Definition 1 *Dependence relationship* (X, Y) *holds for objects* X, Y *if* X *depends on* Y*, but there is no such object* Z *that* X *depends on* Z *and* Z *depends on* Y*.*

Each dependence relation (X, Y) creates an entity called dependency. Objects and dependencies may have properties containing additional information, and they are typed. Type is given as a property according to the following type hierarchy (Fig 2): on the highest level an object can be data, program or feature, a dependency can be read or execute type. On lower levels the type of object is given by a MIME type definition, except for features. Features are special services of the server or user

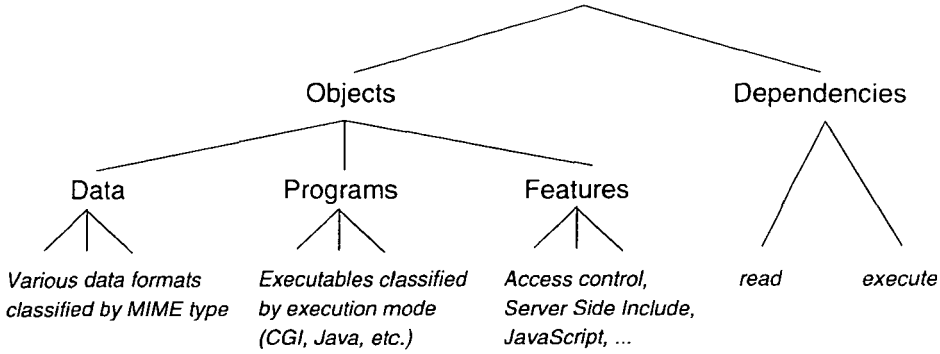


Figure 2: The type hierarchy for the dependence relationship

environment, for example server side include or authorization techniques. In case of dependencies the referencing structure may be given as a property containing the significant HTML or VRML tag, for example an image can be referenced as background (*BACKGROUND*) or inline image (*IMG*).

The reference relationship is defined similarly to the dependence relationship:

Definition 2 *Reference relationship* (X, Y) *holds for objects* X, Y *if* X *contains a hyperlink pointing at* Y.

The difference between dependencies and references is that dependencies show what is needed for an object itself to operate properly, while references show how a set of objects are interconnected to form hypermedia. These two relationships are called together as *portability relationships*.

3.2 Graphical notation and textual representation

The graphical notation for the above defined relationships may be appropriate for human understanding of portability conditions. The notation is explained through an example (Fig 3). This example shows the portability relationships of a simple searchable list (e.g. a hotel list) as a WWW document. Rectangles denote objects and ellipses denote features. References are drawn with dashed lines and dependencies are drawn with solid lines. In the rectangles the name and type of the objects are printed. The dashed ellipse shows the boundaries of the document.

The example document has a starting page containing a logo which is a shared object between several documents, and therefore it is not contained in this document. Searching is done by the search script, called from the home page. The search script requires version 4 of Perl, which is denoted here as a feature of the server environment for simplicity reasons. Another approach would be to denote the Perl program as an external object, and detail other Perl module dependencies as well. This depends on the definition of features. The search script reads the

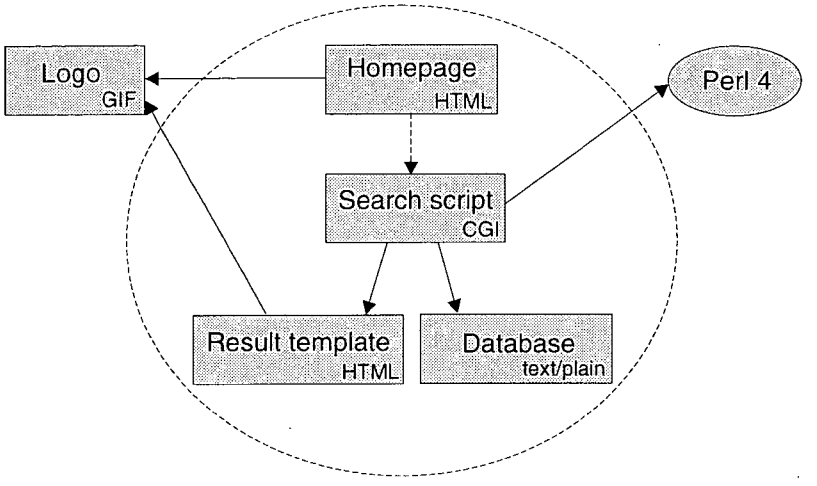


Figure 3: An example for the dependency/reference notation

```
<rdf:RDF>
<rdf:Description about="file:/www/cgi-bin/search_script">
  <PORT:type="CGI" />
  <PORT:dependsOn resource="file:/www/templates/search_result"
     PORT:mode="read" />
  <PORT:dependsOn resource="file:/www/support/index_file"
     PORT:mode="read" />
  <PORT:dependsOn resource="file:/bin/perl4"
     PORT:mode="execute" />
</rdf:Description>
...
</rdf:RDF>
```

Figure 4: RDF representation of dependencies for the search script in Fig 3

database file, and inserts the results of the search into the result template. The generated page contains the logo as well.

Some dependencies are not explicitly drawn on the figure. Each object of type X depends on the feature of handling type X by the server and user environment, but these dependencies are omitted to make the notation clearer. Type names in the boxes implicitly show these feature dependencies. Objects sharing the same dependency can be grouped together with a dashed ellipse to decrease the number of arrows in the graphical representation.

As Fig 3 shows, portability relationships can be interpreted as directed graphs as well. These graphs can be quite complex, and may contain thousands of nodes for a server environment. The graph need not be connected, and it may contain cycles. There are some rules for invalid connections according the type of nodes in the graph, for example an edge from an image to a program is invalid.

Portability relationships fall into the category of structural metadata, and can be described with RDF [17]. RDF (Resource Description Framework) is an effort led by the WWW Consortium to standardize the description of metadata, and coupling metadata with Internet objects. This gives us a machine readable representation of portability metadata applicable for automated document management. Part of the RDF description of the example in Fig 3 is shown in Fig 4.

3.3 Definition of portability requirements

The portability profile is the summary of all portability conditions for the document. It can be calculated by merging all portability relations from various metadata sources and automatic detection processes. For simplicity in further investigations dependency will mean either a dependency or a reference, as the correctness of references are also essential for the correct operation of the whole document.

Definition 3 *Formally a portability profile of document D is*

$$\mathcal{P}_D = \{\, (X,Y) \in \mathcal{P} \mid X \in D \vee \exists Z\colon Z \in D \wedge (Z,X) \in \bar{\mathcal{P}} \,\}$$

where $\mathcal{P}$ contains all described or detected dependencies in the server environment, and $\bar{\mathcal{P}}$ is the transitive closure of $\mathcal{P}$.

The portability profile can be further divided into three subsets:

Definition 4 *Internal dependencies are between objects inside the document:*

$$\mathcal{I}_D = \{\, (X,Y) \in \mathcal{P}_D \mid X \in D \wedge Y \in D \,\}$$

Definition 5 *External dependencies are between document objects and external objects:*

$$\mathcal{E}_D = \{\, (X,Y) \in \mathcal{P}_D \mid X \notin D \vee Y \notin D \,\}$$

Definition 6 *Viewer dependencies are requirements about the services of the user environment:*

$$\mathcal{V}_D = \{\, V \in \mathcal{V} \mid \exists X\colon (X,V) \in \mathcal{P}_D \,\}$$

where $\mathcal{V}$ contains all features that have specific user environment requirements.

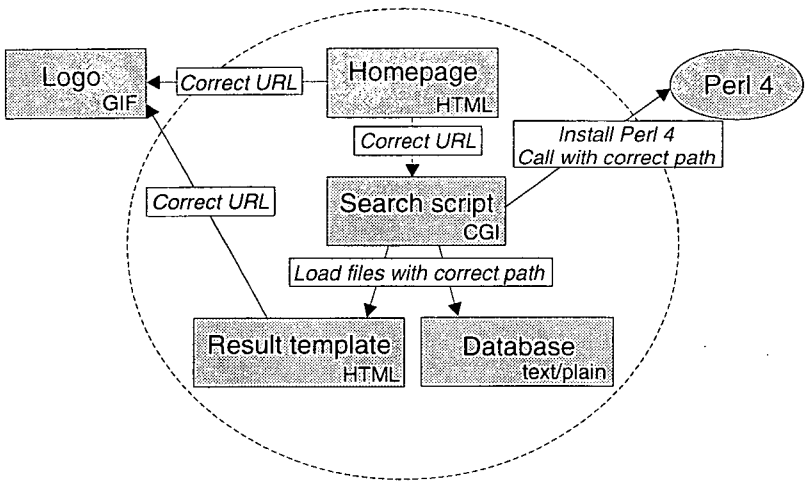


Figure 5: Restoring dependencies (example continued)

Moving/copying a document into a new server environment is formally a process to create correspondance between objects in the original server environment and objects in the new server environment. During this process corresponding objects are found in the new server environment, that are capable to play the same role for the document copy as their pairs in the original server environment.

Definition 7 *Suppose document D' in server environment S' is the copy of document D in server environment S. D' is a* complete mirror *of D if for every (X, Y) in $\mathcal{P}_D$ there is X' and Y' in S' that*

- *X corresponds to X',*
- *Y corresponds to Y',*
- *if X or Y is in D then also X' or Y' is in D'*

and then (X', Y') is in $\mathcal{P}_{D'}$.

However in practice making correspondance is not enough, each dependency has to be operational, has to be working for any user accessing the document. A dependency (X, Y) is *fulfilled* if object X is operational and accessible from object Y. To achieve this the dependency is *tested* and *restored.*

4 Solutions for creating portable documents

A brief list of necessary actions when moving our example document (Fig 3.) is given: the administrator should find the corresponding external objects for the logo

and Perl in the new environment. This could also include installation of Perl, or the decision to use version 5 of Perl to interpret the search script. In Fig 5. the required actions are shown to restore all dependencies. URLs referring to the logo and the search script has to be checked and corrected in the homepage, and in the result template as well. The search script needs two data files and a program file, each loaded by file paths which should be corrected, and execute permissions for the Perl program has to be set. It is obvious from this small example that moving a document means many hardly detectable and easily forgettable tasks for the administrator.

This process can be automated using portability relationships. The automated process contains the following steps:

1. The portability profile is calculated
2. The new place of the document is investigated
3. The document is moved/copied to the new place
4. Each dependency is tested and restored if necessary

Now each step is detailed. The calculation of portability profile needs to rely on portability conditions given by the author, because automatic detection of dependencies is not possible in all cases. It is very easy in an HTML object, but can be nearly impossible in a CGI program written in C where the source code is not available. Therefore the author's contribution is very important, and should be eased with software tools.

The new server environment where the document is to be placed has a set of offered features and documents. These are compared with the external dependency set of the document. The result is a list of missing features or external objects. The administrator of the server environment can install the required features (e.g. Perl 5.0), and match the required external objects to existing objects in the server (e.g. password files). When this is done copying of the document can start.

The document is copied to its new location, and each dependency is tried to be restored. There are three methods for restoration:

- *Changing the object source*: typically used for HTML files, where the URL referring to the depended object can be easily found and corrected in the source.
- *Changing configuration files*: typically used for features (e.g. CGI execution) to enable them in the server configuration.
- *Changing translation tables*: this is a new method and requires that the object accesses the depended object through a translation interface. The object looks for symbolic names in the translation table, which are translated to their path descriptors in the current server environment. Typically used for CGI scripts.

Finally viewer dependencies can provide ways to warn the user automatically about required/missing features in his environment. This can be a listing of required viewer features, or a help to install the missing features.

Generally there are three approaches that can help making WWW documents easily portable: guidelines for authors, document management tools, and application of middleware layers. None of these approaches is exclusive, rather they should be used in conjunction with the others.

4.1 Guidelines for authors

In the first place guidelines can be used to list non-portable technical solutions so that authors can avoid using them. There are also several simple solutions to create very easily portable documents. If relative URLs are used in HTML and VRML, there is no need to restore internal dependencies if the document is moved in one piece.

When using CGI scripts, external data files should be grouped together and referred to with a relative file path. If the script uses external objects, the script should read the location of these objects from a configuration file.

Applets should read their data using the Java built-in resource loading feature and the packaging mechanism.

4.2 Document management tools

The WebPack tool being implemented at SZTAKI is a prototype of a WWW document management tool. It has a notion of document, and maintains a central metadata repository for each document. The central metadata of the document may contain descriptive information about the whole document (e.g. authors, creation date, keywords), and may also contain administrative metadata including portability relations. Metadata can be edited through a graphical user interface, and dependencies in HTML files are automatically detected.

The administrator can move or copy documents inside the server with the WebPack tool. Further useful operations that can be provided by a WWW document management tool are: merging/splitting of documents, removing a document, installing a new document, filtering the contents of the document and verifying the operation of a document.

4.3 Application of middleware layers

A middleware layer could be used to provide standardized communication between the document and the server environment. In this way the immediate dependency of operating system and WWW server features can be lessened to indirect dependency. For example the document calls a software by its standard name, and the middleware layer directs the call to the software in the server environment. Middleware layers can be the solution for automatic configuration of server features as well.

5 Related work

There are several applications that partially support locally manageable WWW documents (e.g. Microsoft's FrontPage, Macromedia's Dreamweaver). These usually handle the web server as a whole and automatically adjust URLs in HTML files when it is needed. Netscape Composer's publishing functionality automatically updates the links when the page is published. However these solutions are restricted to HTML files, and lack a consistent approach to all possible portability problems on the Web.

There are two categories of public domain tools in connection with our topic: HTML integrity test tools, and mirroring tools. Mirroring tools replicate the files of a document on a remote server via HTTP or FTP protocol. However it is not always possible to retrieve all needed files in this way because of access and authentication problems, furthermore these programs do not explore and restore spoilt dependencies. For the latter HTML integrity test tools can be used. These traverse the hyperlinks in HTML files and report dangling URLs.

An organized standardization effort of IETF in this area is WebDAV (World Wide Web Distributed Authoring and Versioning) [18] which aims at "defining the HTTP extensions necessary to enable distributed web authoring tools to be broadly interoperable, while supporting user needs". In this respect WebDAV will support remote management and authoring of WWW pages, but currently it is only near the end of the standardization phase.

Location transparent naming is a central problem for portability. The CNRI handle system [10] and PURLs (Persistent URLs) [11] are two experiments for location transparent naming on the Internet. The problem with both systems is that they provide a flat naming scheme for objects, and thus will not be able to cope with the millions of existing WWW pages.

Object request brokers and distributed object management may give a long term solution for powerful and portable hypermedia documents. The Object Management Group's Object Management Architecture and Common Object Request Broker Architecture (CORBA [13]) will provide a very general framework for network objects including naming services. These network objects could encompass documents or parts of documents. However as a short and middle term solution the Internet community needs the tools outlined in Section 4.

6 Summary

The complex issues of portability for hypermedia documents on Internet were summarized, and a unified approach to handle portability problems was given. This approach contains a formal description technique for portability dependencies, and a method to match and restore these dependencies. Using this approach the exchange of complex WWW documents could become as simple as the exchange of documents in Microsoft's Word format. The WebPack framework serves as a testbed for the implementation of technical solutions based on the theoretical in-

vestigations.

Acknowledgment I would like to thank Róbert László his work in the implementation of the WebPack toolkit.

References

[1] About the World Wide Web, URL: http://www.w3.org/pub/WWW/WWW/

[2] WWW Names and Addresses, URIs, URLs, URNs, URL: http://www.w3.org/hypertext/ WWW/Addressing/Addressing.html

[3] Hypertext Markup Language, URL: http://www.w3.org/hypertext/ WWW/MarkUp/MarkUp.html

[4] Hypertext Transfer Protocol, URL: http://www.w3.org/hypertext/WWW/ Protocols/Overview.html

[5] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738

[6] Rob McCool: The CGI Specification, URL: http://hoohoo.ncsa.uiuc.edu/ cgi/interface.html

[7] Java Technology Home page, URL: http://java.sun.com

[8] JavaScript resources, URL: http://developer.netscape.com/tech/javascript /resources.html

[9] VRML97 International Standard, URL: http://www.vrml.org/Specifications

[10] The CNRI Handle System, URL: http://www.handle.net/

[11] Persistent URLs, URL: http://purl.oclc.org/

[12] Common Ground Digital Paper, URL: http://www.hummingbird.com/cg/

[13] CORBA (Common Object Request Broker Architecture) URL: http://www. omg.org/corba/

[14] Reginald Ferber: Hypermedia and Metadata, 2nd DELOS Workshop, October 1996, Bad Honneff, Germany, URL: http://www.darmstadt.gmd.de/ ~ferber/delos/ws2/frame/frame.html

[15] Stuart Weibel, Jean Godby, Eric Miller and Ron Daniel: OCLC/NCSA Metadata Workshop Report, URL: http://www.oclc.org:5046/oclc/ research/conferences /metadata/dublin_core_report.html

[16] Carl Lagoze, Clifford A. Lynch, Ron Daniel Jr.: The Warwick Framework - A Container Architecture for Aggregating Sets of Metadata, Cornell Computer Science Technical Report TR95-1558

[17] Resource Description Framework, URL: http://www.w3.org/Metadata/rdf

[18] IETF WebDAV Working Group, URL: http://www.ics.uci.edu/~ejw/ authoring

[19] L. Kovács, A. Micsik: "Portable Hypermedia: a New Format for WWW Documents", SZTAKI Technical Report TR97-1

[20] L. Gulyás, L. Kovács, A. Micsik, L. Tersztenyák: Personalized Home Pages - A Working Environment on the World Wide Web, *to appear in:* IFIP'98 World Computer Congress, Vienna-Budapest, September 1998

[21] L. Kovács, A. Micsik: Replication within Distributed Digital Document Libraries. Proceedings of the 8th ERCIM Database Research Group Workshop on Database Issues and Infrastructure in Cooperative Information Systems, Trondheim, Norway, 1995

[22] L. Kovács, A. Micsik, G. Schermann: An Environment for Mirroring Hypermedia Documents, JENC 7, Budapest, May 13-16 1996.

Acta Cybernetica 14 (1999) 403-418.

A parallelized sequential random search global optimization algorithm

P.M. Ortigosa, J. Balogh, I. García *

Abstract

This work deals with a stochastic global optimization algorithm, called CRS (Controlled Random Search), which originally was devised as a sequential algorithm. Our work is intended to analyze the degree of parallelism that can be introduced into CRS and to propose a new refined parallel CRS algorithm (RPCRS). As a first stage, evaluations of RPCRS were carried out by simulating parallel implementations. The degree of parallelism of RPCRS is controlled by a user given parameter whose value must be tuned to the size of the parallel computer system. It will be shown that the greater the degree of parallelism is the better the performance of the sequential and parallel executions are.

Keywords: Parallel algorithm, Distributed Processing, Random Search, Global Optimization.

1 Introduction

The generic global optimization problem can be described as:

$$\min f(s),\ s \in S \subset R^n \tag{1}$$

where the objective function, $f(s)$, is a real valued continuous nonlinear function on S and the search domain, S, is a compact body. Under these conditions it is known that the optimal solution value:

$$f^* \equiv \min f(s), s \in S \tag{2}$$

exists and is attained; i.e. the set:

$$S^* \equiv \{s \in \dot{S} : f(s) = f^*\} \neq \emptyset \tag{3}$$

Two general models of Global Optimization methods exist: Deterministic methods which require a certain mathematical structure and Stochastic methods which

*This work was supported by the Tempus Program, by the Consejeria de Educación de la Junta de Andalucía (07/FSC/MDM) and by the Ministry of Education of Spain (CICYTTIC96-1125-C03-03).

are based on the random generation of feasible trial points and nonlinear local optimization procedures. A profound discussion on the classification of methods can be found in Törn and Zilinskas [15] and for a complete and rigorous mathematical description of global optimization methods, both deterministic and stochastic approaches, the reader may consult the Handbook of Global Optimization [6].

On the other hand, there exists a question which frequently arises when a practical Global Optimization problem has to be solved: *which kind of method, or which particular algorithm, may be more appropriate to solve a particular problem?*. The answer could only be obtained from a deep analysis of the problem at hand. A wide discussion on this subject can be found in [5], where relations between the problem, its modeling and properties, and global optimization methods are studied. Roughly speaking, it could be said that deterministic methods may be more efficient than the stochastic ones, when an analytical expression of f, its derivatives, bounds and useful properties are available. However, when f is a black-box function deterministic methods cannot be applied. In contrast, stochastic methods do not require any specific structure of f, only a computational procedure to obtain the value of the function at any location $s \in S$ [5] is needed. So, most optimization problems can be solved by stochastic global optimization techniques.

For computationally expensive functions, stochastic global optimization methods have shown to be very useful because of, compared to deterministic methods, fewer function evaluations are needed to obtain the solution of (1). In addition, stochastic methods can be applied to problems where the objective function is not continuous nor differentiable and only a tool for evaluating the function at any location is required.

For most of the functions, global optimization is a NP hard problem. For this reason, the global optimization problem is a suitable candidate for the use of supercomputers, mainly for those functions whose evaluation is computationally expensive.

This paper will only deal with a stochastic global optimization algorithm called CRS and a new parallel version of CRS. CRS (Controlled Random Search) algorithm was introduced by Price [9, 10, 11]. It is based on clustering techniques and has proved to be very reliable and computationally inexpensive. In [2] a parallel version of CRS (PCRS) was applied to efficiently solve a global optimization problem coming from the image processing field, whose objective function were computationally very expensive.

The aim of this work is to describe and evaluate a new refined version of PCRS, called RPCRS. It was originally devised to be executed on parallel multicomputer system, but it will be also shown that RPCRS outperforms CRS even when it is run on a single processor system. Our study only covers analysis of the speed up of RPCRS as compared to the original sequential CRS algorithm. Our analysis is only based on empirical results obtained from experimental executions. Although a wide set of standard test functions was used to validate our results, this work does not provide any theoretical support to demonstrate that the same results can be obtained using other functions.

Two different kinds of experiments were carried out for analyzing the speed up

of RPCRS: those oriented to highlight advantages of its parallel nature and those intended to show its capability for being executed on a parallel computer system.

This paper has been organized as follows: Section 2 describes the CRS algorithm and its parallel version RPCRS. Section 3 is devoted to show experimental results intended to evaluate the speed up of RPCRS compared to CRS, as a function of a control parameter which determines the degree of parallelism. Finally, in Section 4, numerical results of parallel executions of RPCRS, on a CRAY T3D using up to 16 processor elements, will be shown.

2 RPCRS, a parallel version of the CRS algorithm

The goal of this section is to describe the RPCRS algorithm; a refined version of PCRS (Parallel Controlled Random Search) algorithm proposed in [2]. RPCRS is a parallel algorithm based on the Controlled Random Search (CRS) algorithm of Price [9, 10, 11]. Some parallel approaches of the original CRS algorithm have been proposed and evaluated using several models of parallel computers and strategies [1, 3, 4, 7, 12, 14, 16]. Our proposal only makes small modifications to the original sequential version of CRS. These modifications are aimed at increasing the degree of parallelism of CRS by creating a pile of work to feed the set of processors of a parallel computer. RPCRS will allow to evaluate the objective function at several trial points, simultaneously. Nevertheless, the general strategy used in CRS remains in our parallel version. For the sake of clarity, description of CRS will precede to RPCRS algorithm.

The Controlled Random Search algorithm, proposed by Price, is a simple and direct procedure for global optimization, applicable both to unconstrained and constrained optimization problems [9, 10, 11]. In this work, it is assumed that the global optimization problem to be solved is that described by (1), where S is a hyper-rectangle. Due to its simplicity, CRS has been used to solve many practical problems but it has not been very popular among researchers on the theory of Global Optimization because no analytical property can be derived.

CRS starts by evaluating the objective function at N trial points randomly chosen over S, (**initialize** step of Algorithm 2.1). Coordinates and the corresponding value of the objective function, for the set of N trial points, are stored in an array $R = R^0, \ldots, R^N$. The worst and besttrial points in R (R^W, R^B) are then computed at the **update** step. New trial points ($\overline{P}$) are selected and evaluated, at the **generate** step. The algorithm iteratively executes the **update** and **generate** steps until stopping criteria are reached.

At the **generate** step two different trial points are computed; primary and secondary trial points. Both kinds of trial points are defined in terms of the configuration of a subset of $n+1$ ($R^{j_0}, \ldots, R^{j_n}$) trial points. R^{j_i}, $(i = 0, \ldots, n)$ are randomly selected from the current set of N points stored at R (CRS is considered the first algorithm which uses a population of points). Primary points are generated in a Nelder-Mead fashion [8] by mirroring a point (R^{j_n}) over the centroid, $\overline{G}$, of the remaining subset of points ($R^{j_0}, \ldots, R^{j_{n-1}}$). In contrast, location of a sec-

Algorithm 2.1
Begin **CRS**$(f, S, N, NF_{max}, \epsilon)$
initialize:
$Iter = 0$; $D_{max} = \epsilon + 0.1$; $ns = 0$
Select at random a set, R, of N trial points R^i; $(0 \leq i < N)$
Compute $f(R^i)$; $(0 \leq i < N)$
while $(D_{max} > \epsilon$ *OR* $f(R^W) - f(R^B) < \delta$ *OR* $Iter < NF_{max})$
update: *Determine the trial points* W *and* B *such that*
$f(R^W) \geq f(R^i) \geq f(R^B)$; $(0 \leq i < N)$
$Iter = Iter + 1$
Begin **generate:**
Select randomly $n+1$ *points,* (R^{j_i}), *from the set* R
$\overline{G} = \sum_{i=0}^{n-1} R^{j_i}/n$
$\overline{P} = 2 \times \overline{G} - R^{j_n}$. # **Primary points**
if $\overline{P} \in S$ *AND* $f(\overline{P}) < f(R^W)$
$R^W = \overline{P}$; $ns = ns + 1$
else if $RS = ns/Iter < 0.5$ # **Rate of Success Test**
$\overline{P} = (\overline{G} + R^{j_n})/2$ # **Secondary points**
$Iter = Iter + 1$
if $f(\overline{P}) < f(R^W)$
$R^W = \overline{P}$; $ns = ns + 1$
End **generate**
End **while**
Return $\{R^B$ *and* $f(R^B)\}$; # $f(R^B) \leq f(R^i)$; $(0 \leq i < N)$
End **CRS**

ondary point is the middle point between R^{j_n} and the centroid $\overline{G}$. While primary points are intended to keep the search space as wide as possible (global search), secondary points are conducive to convergence (local search). Secondary points are only computed if the current primary trial point fails and the rate of success (RS) in finding smaller values of R^W is bellow 50%. This general procedure may be modified in a variety of ways, our version of CRS is detailed at Algorithm 2.1, where stopping criteria are based on (i) the value of the maximum distance between any two points in the set R ($D_{max} = \max\{d(R^i, R^j);\ \forall\, 0 \leq i, j < N;\ i \neq j\} \leq \epsilon$), (ii) the range of $f(R^i)$, i.e. $f(R^W) - f(R^B) < \delta$; where $f(R^W) = \max\{f(R^i)\}$; $f(R^B) = \min\{f(R^i)\}$; and (iii) the number of function evaluations NF_{max}. Condition (i) ensures that all the trial points are located in a small cluster, condition (ii) allows the algorithm to stop even when the trial points are a long distance apart but the values of the function for all the trial points are almost equal one to each other; this condition is useful for functions with several global optima. Finally, condition (iii) will permit to leave the process in the case that algorithm does not

converge.

It can be seen that CRS is a highly sequential algorithm, because every new trial point, $\overline{P}$, is generated from a subset of the current set R of N trial points. This current set of points consists of the best points found along the iterative procedure. However, provided that $R^0, \ldots, R^{N-1}$ can be simultaneously generated and $f(R^0), \ldots, f(R^{N-1})$ concurrently computed, the algorithm exhibits some degree of parallelism at the **initialize** stage.

In order to increase the degree of parallelism of CRS, the following strategy has been introduced: After the **initialize** stage, using the same initial set (R) of N trial points, a set of b primary points are generated and saved into a FIFO buffer, $A = A^0, \ldots, A^{b-1}$. After computing $f(A^0)$, R^W will be replaced by A^0 iff $f(A^0) < f(R^W)$. A^0 is removed from the buffer and a new point is obtained by the **generate** procedure and saved at the end of the buffer FIFO. The only difference with Algorithm 2.1 is that a set of b trial points is always ready to be evaluated, so the degree of parallelism is increased.

The best strategy for implementing this kind of parallel algorithms is a centralized model, where a master-worker communication scheme is applied. In our model, the master processor executes the optimization algorithm and provides a set of trial points to the worker processors. worker processors only evaluate the objective function at the trial points supplied by the master processor and after every evaluation of the function they send the result back to the master processor [4]. García et al [2, 4] have implemented a similar strategy using a fully asynchronous model where the master processor does not start to generate primary or secondary trial points until the initial sample set of trial points has been evaluated. At any time worker processors keep information of a single trial point. Although this approach is fully asynchronous, several worker processors frequently may remain in a idle state waiting for the master processor to provide a new trial point.

This drawback could be solved if worker processors always keep in a buffer a set of trial points to be evaluated. So, when a worker processor finishes an evaluation, it sends the result back to the master and goes on evaluating a new trial point stored in its local buffer. Our parallel implementation of CRS (RPCRS) consists of two different processes: **Master_RPCRS** (Algorithm 2.2) and **Worker_RPCRS** (Algorithm 2.3).

The **Master_RPCRS** process consists of three different stages: (i) the **initialize** stage where the set R of N trial points are randomly chosen over S, (ii) a stage where $b = NP \times npoints$ primary trial points are computed and (iii) the convergence loop where primary or secondary trial points are generated following a strategy similar to Algorithm 2.1.

After **initialize** step, master processor cyclically distributes all the N trial points among worker processors. If the number of worker processors (NP) were greater than N, master processor would generate NP trial points and after receiving their function values from worker processors, it would only save the best N trial points at R. Then, master processor calculates b primary trial points and sends $npoints$ to each worker processor. Points $\overline{P}$ are computed using b different $\overline{G}$ and R^{j_n}.

Algorithm 2.2
Begin **Master_RPCRS**$(f, S, N, NP, npoints, NF_{max}, \epsilon)$
initialize:
$Iter = 0$; $D_{max} = \epsilon + 0.1$; $ns = 0$
Select at random a set, R, of N trial points R^i; $(0 \leq i < N)$
SEND N/NP *trial points from R to each worker processor*
RECEIVE N *function values from the* NP *processors*
do $j = 1 : NP$
do $k = 1 : npoints$ # $b = NP \times npoints$
$Iter = Iter + 1$
$\overline{G}^k = \sum_{i=0}^{n-1} R^{j_i}/n$
$\overline{P}^k = 2 \times \overline{G^k} - R^{j_n}$. # **Primary points**
SEND $\overline{P}^k$ *to processor* j; $(k = 1, \ldots, npoints)$.
while $(D_{max} > \epsilon$ *OR* $f(R^W) - f(R^B) < \delta$ *OR* $Iter < NF_{max})$
update: *Determine the trial points W and B such that*
$f(R^W) \geq f(R^i) \geq f(R^B)$ $(0 \leq i < N)$
$Iter = Iter + 1$
$\overline{P}_{new}$ = **gen_trial()**
RECEIVE $f(\overline{P})$ *from processor idp* # $(1 \leq idp \leq NP)$
SEND $\overline{P}_{new}$ *to processor idp*
if $f(\overline{P}) < f(R^W)$
$R^W = \overline{P}$; $ns = ns + 1$
End **while**
Return $\{R^B$ *and* $f(R^B)\}$; # $f(R^B) \leq f(R^i)$; $(0 \leq i < N)$
End **Master_RPCRS**

In the convergence loop the greatest value of the function $(f(R^W))$ in the set $R^0, \ldots, R^{N-1}$ is determined. Also a new $\overline{P}$ (named $\overline{P}_{new}$) is computed in **gen_trial()** procedure. In this procedure the algorithm will generate a primary or a secondary trial point following the same strategy of CRS. Then, master processor waits for the arrival of a new value of the objective function from any worker processor and immediately sends a new trial point $\overline{P}_{new}$ to this worker processor. After that, master processor checks if the received trial point is accepted or not and decides if the next trial point should be a primary or a secondary trial point.

Worker_RPCRS process consists of an initial stage where worker processor receives N/NP trial points from master processor, evaluates them and returns the values of the function back to master processor. Then, worker processor receives *npoints* to be evaluated. They are stored in a FIFO buffer, A. When a worker processor has evaluated a trial point it sends the value of the function back to the master processor and checks for the arrival of a new trial point. If a new point

Algorithm 2.3
Begin **Worker_RPCRS**$(f, S, N, NP, npoints)$
RECEIVE *N/NP trial points and store in a FIFO A*
Compute $f(A^i)$; $1 \leq i \leq N/NP$
SEND *N/NP values of the function ($f(A^i)$) to the master processor*
RECEIVE *npoints from the master and store them into* $A^{first}, \ldots, A^{last}$
while *(***Master_RPCRS** *is working)*
while *there are stored points to evaluate* ($A \neq \emptyset$)
Evaluate $f(A^{first})$
SEND $f(A^{first})$ *to master processor*
if *a new point has arrived from master*
RECEIVE A^{last}
end **while**
wait for a new point from master processor
RECEIVE A^{last}
End **while**
End **Worker_RPCRS**

has arrived, worker processor reads the point and pushes it into the FIFO buffer. Otherwise, worker processor goes on evaluating the next point of its buffer. If the FIFO buffer $A = \emptyset$ and master processor is still working, worker processor has to wait for a new trial point from the master processor.

Using this strategy, idle time of worker processors is reduced (even eliminated), and in addition communication overhead is decreased because communications and computations are overlapped.

3 Evaluation of RPCRS on a uniprocessor environment

In order to hide the set of problems associated to parallel implementations, such as communication overhead or bottlenecks due to intensive communications, a machine independent evaluation of RPCRS has been realized; i.e. in this section executions of RPCRS were carried out on a uniprocessor system and performance was measured versus the number of function evaluations computed during the execution of RPCRS. The goal of this analysis consists of determining the behavior of RPCRS with respect to CRS as a function of the buffer size (b).

A set of twenty two test functions has been used to check convergence and parallel performance of RPCRS. Due to the strong stochastic component of this algorithm, the number of function evaluations carried out by RPCRS depends on the particular execution. For this reason, the algorithm has been executed 100 times

for every value of the setting parameters, obtaining a significantly statistic sample. From this data set, average value of the number of function evaluations and the corresponding confidence intervals (95%) were computed (see [13]). Setting lower and upper limits to a statistic implies that the probability of an interval covering the mean is 0.95 or, expressed in another way, that on the average 95 out of 100 confidence intervals similarly obtained would cover the mean.

Table 1: Results of RPCRS for Goldstein/Price and Shekel10 test functions.(1) and (2) mean 98% and 97% of success were obtained respectively.

Goldstein/Price Test Function						
	Cluster Size = $25 \times n$		Cluster Size = $50 \times n$		Cluster Size = $100 \times n$	
b	NoFE	Conf.Int.	NoFE	Conf.Int.	NoFE	Conf.Int.
1	(1)1622	[1604,1640]	3254	[3230,3278]	6505	[6467,6544]
2	1650	[1632,1667]	3261	[3238,3284]	6511	[6479,6543]
3	1633	[1609,1657]	3261	[3238,3284]	6502	[6473,6532]
4	1649	[1630,1669]	3256	[3229,3283]	6501	[6464,6539]
8	1631	[1612,1649]	3235	[3207,3264]	6520	[6483,6558]
16	1599	[1575,1623]	3212	[3186,3237]	6431	[6394,6468]
32	1606	[1574,1637]	3146	[3112,3180]	6340	[6301,6378]
64	1706	[1674,1739]	3149	[3112,3186]	6229	[6195,6264]
Shekel10 Test Function						
1	5310	[5262,5357]	10614	[10562,10666]	21454	[21367,21540]
2	(2)5315	[5260,5370]	10645	[10592,10698]	21543	[21475,21611]
3	5326	[5283,5369]	10620	[10562,10678]	21523	[21433,21613]
4	5335	[5295,5375]	10576	[10506,10646]	21498	[21425,21571]
8	5279	[5229,5329]	10575	[10509,10642]	21472	[21390,21553]
16	5161	[5117,5206]	10477	[10409,10545]	21452	[21371,21532]
32	5049	[4945,5152]	10237	[10171,10303]	21066	[20987,21145]
64	5059	[4905,5213]	9784	[9728,9839]	20660	[20577,20743]

In order to facilitate analysis of the behavior of RPCRS, in a first set of experimental tests, only six test functions were used: Goldstein/Price, Hartman3, Hartman6, Shekel5, Shekel7 and Shekel10 [15].

Two of the RPCRS's input parameters which play an important role in the performance evaluation are: the number of trial points N defining the cluster of trial points and the size b of the buffer. During the first set of tests, performance evaluation has been made as a function of both N and b. N has been established as a function of the dimension of the problem n, so values for N in our performance evaluation were $N = 25 \times n$, $50 \times n$, $75 \times n$ and $100 \times n$. Values for the buffer size b were 1, 2, 3, 4, 8, 16, 32, 64. It must be pointed out that for $b = 1$, RPCRS algorithm matches to the original sequential version of CRS algorithm. So,

performance evaluation will be realized by comparing RPCRS(b) to RPCRS($b = 1$). For all the executions of RPCRS, the parameters used in the stopping criteria were: $\epsilon = 10^{-5}$, $\delta = 10^{-5}$ and $NF_{max} = 10^6$.

In Table 1 numerical results obtained from the evaluation of two test functions, using several values of b and N, are given. In this table average values of the number of function evaluations (NoFE) and the corresponding confidence intervals (Conf.Int.) for a sample of hundred executions of RPCRS, are shown.

From Table 1, it must be noticed that for the smallest value of the cluster size ($25 \times n$) the percentage of success of RPCRS were not always 100% (see notes (1) and (2)). Therefore, bigger values of the cluster size N must be chosen to ensure the convergence of the algorithm. It can be seen in Table 1 that the number of function evaluations tends to decrease when the cluster size N grows, though this tendency can not be observed for Goldstein/Price Function with the smallest cluster size value ($25 \times n$)

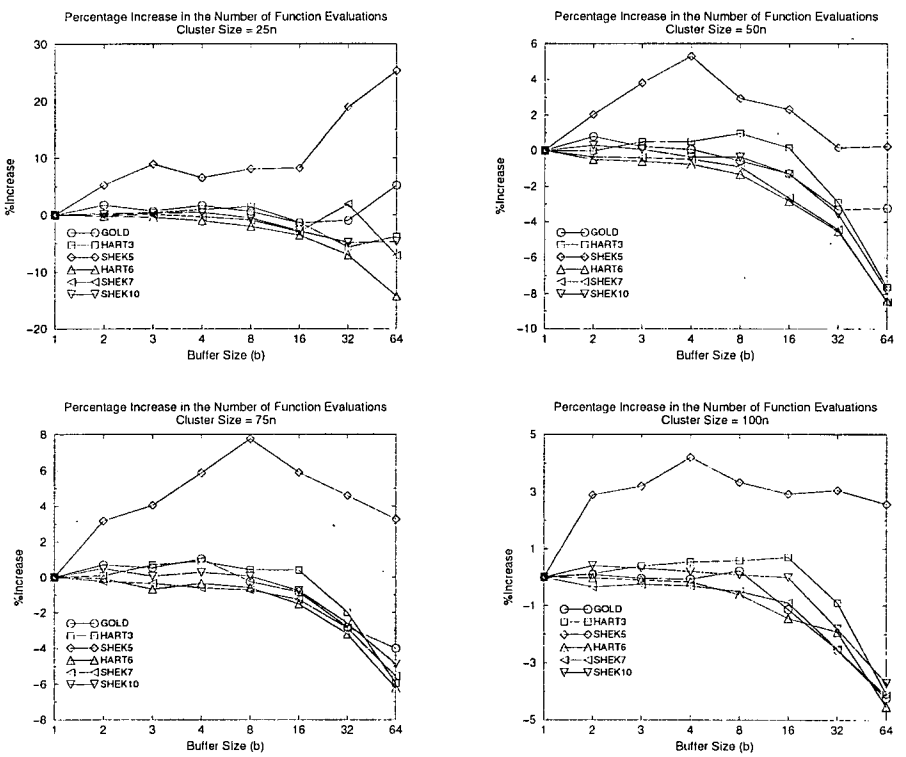


Figure 1: Percentage of increase in the number of function evaluations ($NoFE$) for several values of the cluster size (N): % Increase = $\frac{NEval(b) - NEval(b=1)}{NEval(1)} \times 100$.

Figure 1 shows results of RPCRS over the set of six test functions for the four values of cluster size. Due to the average values of the number of function evaluations range between 1000 and 40000, we decided not to draw the number of function evaluation versus the buffer size b. Instead of this, in Figure 1, the percentage of increase (%Increase) in the number of function evaluations with respect to the original case ($b = 1$), ($\frac{NEval(b)-NEval(b=1)}{NEval(1)} \times 100$), has been drawn.

So Figure 1 shows the percentage of increase (or decrease) for several different values of the cluster size. For Shekel 5 test function, a positive %Increase were always obtained, though this increment in the number of function evaluations diminishes as the cluster size increases. For the remaining test functions, it can be seen that %Increase varies just a little bit because the average values of the number of function evaluations remains almost constant, with respect to b, when the buffer size is small ($1 \leq b \leq 8$). For bigger buffer sizes the %Increase tends to be more negative (e.g. Hartman6). Only for Goldstein/Price test function with a cluster size $N = 25 \times n$, the %Increase is positive for a buffer size $b = 64$. This is due to the number of points stored in the buffer is relatively large compared with the cluster size. Goldstein/Price test function is two-dimensional and therefore the cluster size in this case is smaller, $25 \times n = 50$, than the buffer size ($b = 64$); i.e. $\frac{N}{b} < 1$.

Results from this set of experiments seems to show that: (i) using a cluster size large enough, convergence to the global optimum is ensured and (ii) for a value of the buffer size smaller than the cluster size but greater than 8 the computational cost of RPCRS diminishes or remains similar to the original CRS (RPCRS($b = 1$)).

In a second set of experiments, performance evaluation of RPCRS was made for a wider set of test functions (see appendix for a detail description of these test functions). In this case the cluster size was always $N = 100 \times n$, ensuring that for all the test functions $N > b$. This new set of test functions includes all the functions previously used and seventeen additional functions. These functions have been chosen in such a way that they are defined over several different domains of definition, $S \subset R^n$, where S ranges from $[-1, 1]$ to $[-600, 600]$ and n from 1 to 10; the number of global optima varies from 1 to > 10 and at least one of the functions has more than 1000 local optima. For all the functions, 100% of success in finding the optimal solution was obtained by RPCRS.

In Figure 2, performance evaluation of RPCRS, for the set of 22 test functions, is shown. In this graph, X-axe represents the index of the test function. The functions have been sorted by the increasing number of function evaluations needed to reach the global solution when $b = 1$ ($NEval(b = 1)$). For each function, results for all the values of the buffer size (b) are displayed in a vertical straight line. Roughly speaking it can be said that when the number of function evaluations is big enough the performance of RPCRS is best than that of CRS; i.e. $NEval(b) < NEval(b = 1)$ for most of the values of b. For test functions with a computational burden not too strong, the performance of the algorithm depending of the value of the buffer size, has more fluctuations. Anyway it can be seen that only for 5 functions over the set of 22 test functions, RPCRS performs worst than CRS; i.e $NEval(b) > NEval(b = 1)$.

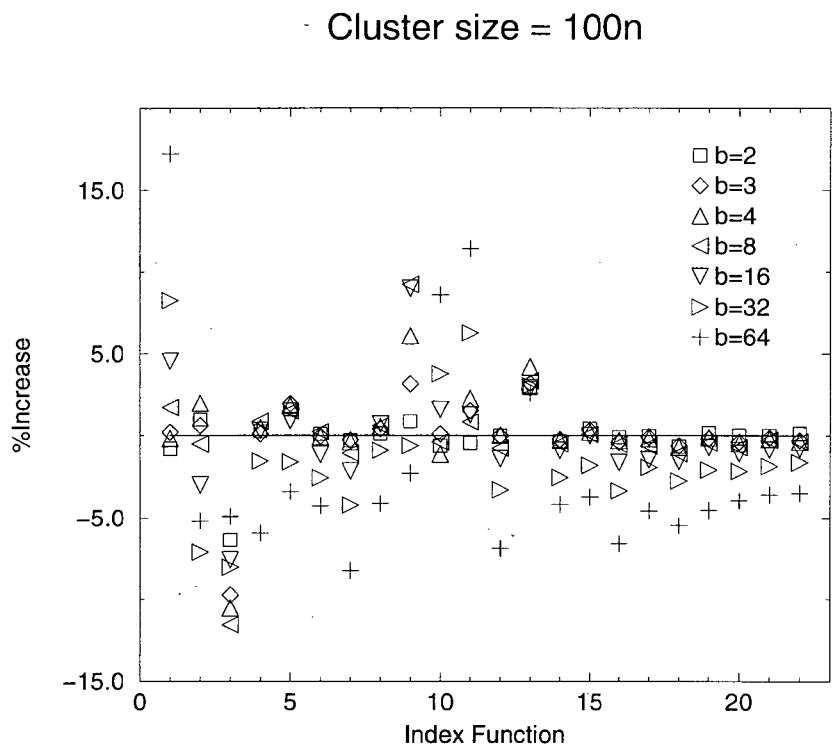


Figure 2: Percentage of increase in the number of function evaluations for several values of the buffer size (b) and $N = 100 \times n$. % Increase $= \frac{NEval(b) - NEval(b=1)}{NEval(1)} \times 100$.

In general, it can be said that the parallelism introduced at CRS for building RPCRS does not strongly disturb its performance characteristic and for hard functions, requiring a lot of computational resources, RPCRS outperforms CRS. Nevertheless, no theoretical proof of these results has been still studied. In the next section, results of the performance evaluation of RPCRS on a parallel system (Cray T3D) using up to 16 processors are presented.

4 Performance evaluations on a parallel system

In order to analyze the behavior and the performance of the parallel program, we have chosen a cluster size of $100 \times n$. In our experiments buffer sizes $b = 16$ and $b = 64$ were used. Though our asynchronous parallel program was designed with the capability of overlapping computations and communications, the algorithm needs

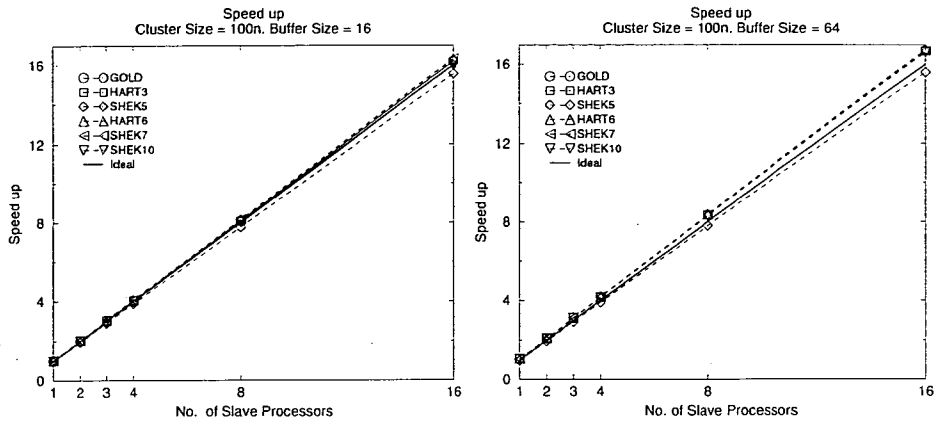


Figure 3: Speedup of the parallel executions of RPCRS with respect to the sequential case. Speed-up $=t(RPCRS(b = 1, p = 1))/t(RPCRS(b, p))$. Delay = 0.03 sec.

the computational cost of the test functions to be greater than the communication time required to exchange data among master and worker processors. If computational cost of the objective function is small enough, bottlenecks would appear at master processor. But for these unexpensive functions it would not be necessary the use of a parallel system. So, in our evaluations of the parallel performance of RPCRS, it was simulated that test functions have the same computational cost by introducing an additional time delay into the function evaluation. Particularly, effects of delays: 0.003 sec. and 0.03 sec. have been analyzed for a set of six test functions.

Figure 3 shows the values of the speedup obtained when the execution time for the parallel executions is compared to the execution time obtained by the original sequential algorithm ($CRS = RPCRS(b = 1, p = 1)$); i.e. Speed-up $=t(RPCRS(b = 1, p = 1)/t(RPCRS(b, p))$, where p is the number of worker processors. From results at Figure 3, it might seem that we have implemented a marvelous parallel program because of a speedup over linear is most of times obtained. Nevertheless, these super speedups are due to the property that the number of function evaluations carried out by RPCRS algorithm is lesser for buffer sizes $b = 16$ or $b = 64$ than for a buffer size $b = 1$ (see Figure 1).

Figure 4 shows values of the speedup obtained when execution time for the parallel program is compared to execution time obtained in the sequential case, but in this case using the same value of the buffer size for both sequential and parallel executions; i.e. Speed-up $= t(RPCRS(b, p = 1)/t(RPCRS(b, p))$. It can be seen that in this case an almost linear speedup has been obtained for all the functions, but no super speedups. These speedups are closer to the linear speedup

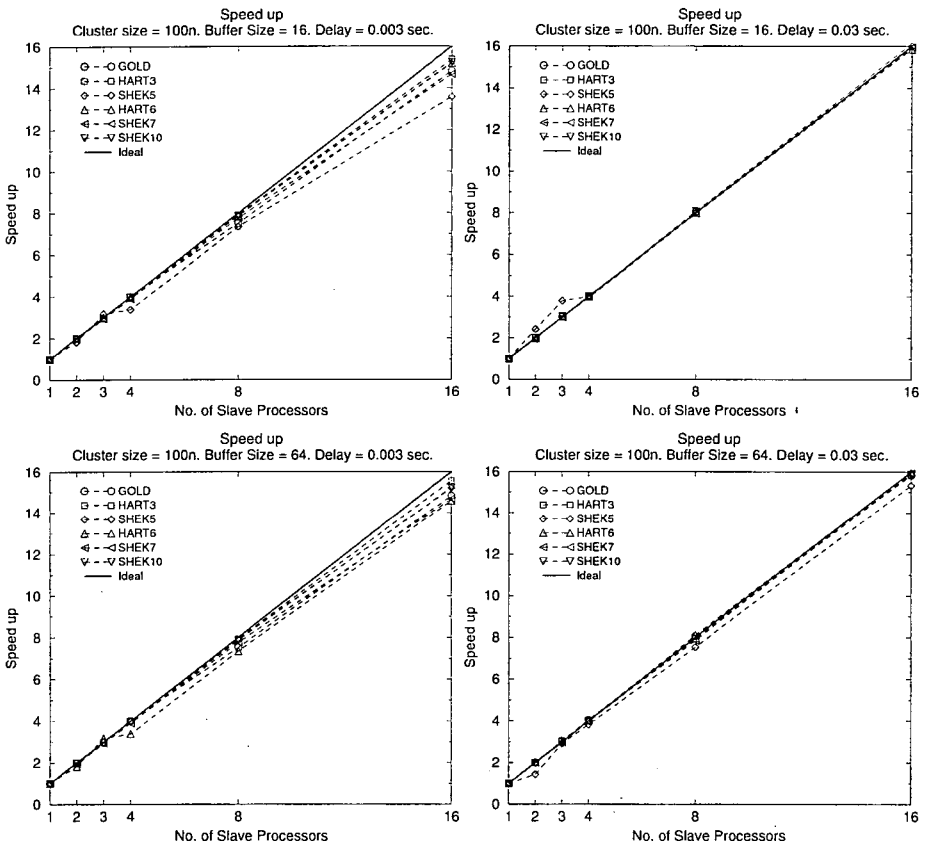


Figure 4: Speedup of the parallel version with respect to the sequential case. Speed-up $= t(RPCRS(b, p = 1))/t(RPCRS(b, p))$.

for a delay of 0.03 sec. than for a delay of 0.003 sec.

5 Conclusions and future work

In this work a parallel implementation of CRS, called RPCRS, has been described and evaluated. It has been shown that RPCRS is computationally cheaper than CRS and in addition it is easy to implement on a real parallel or distributed computing system an asynchronous model.

Although a wide set of standard test functions were used to validate that RPCRS outperforms to CRS, no theoretical support exists behind this work to demonstrate that our results will be the same for any function. Our future work will be aimed

at obtaining a better understood of this results and a mathematical proof (if any).

Appendix: Description of the test functions

F : Index of the Function.
D : Search domain.
$f(x^*)$: Global minimum value of the function.
M : Number of global plus local minima of the function.

Table 2: Description of the test functions.

F	Function f(x)	D	$f(x^*)$	M
1	Three hump camel back	$[-5,5]^2$	0.0	3
2	$(x_1-5)^2-(x_2-10)^2$ if $x_1 \leq 10$ $(x_1-15)^2-(x_2-10)^2$ otherwise	$[0,20]^2$	0.0	2
3	Six hump camel back	$[-2.5,2.5]^2$	-1.0316	6
4	Booth	$[-5,5]^2$	0.0	1
5	Levy 13	$[-10,10]^2$	0.0	≥ 1
6	Goldstein / Price	$[-2,2]^2$	3.0	3
7	Shperical 2 $=\sqrt{\sum_i^2 x_i^2}$	$[-1,1]^2$	0.0	1
8	Hartman 3	$[0,1]^3$	-3.862782	≥ 3
9	Beale	$[-5,5]^2$	0.0	> 4
10	Levy 3	$[-10,10]^2$	-176.54	≥ 9
11	Griewank	$[-600,600]^2$	0.0	≥ 10
12	Shperical 3 $=\sqrt{\sum_i^3 x_i^2}$	$[-1,1]^3$	0.0	1
13	Shekel 5	$[-10,10]^4$	-10.15320	≥ 4
14	Shekel 7	$[-10,10]^4$	-10.40294	≥ 4
15	Shekel 10	$[-10,10]^4$	-10.53641	≥ 4
16	Shperical 4 $=\sqrt{\sum_i^4 x_i^2}$	$[-1,1]^4$	0.0	1
17	Hartman 6	$[0,1]^6$	-3.322828	≥ 6
18	Shperical 5 $=\sqrt{\sum_i^5 x_i^2}$	$[-1,1]^5$	0.0	1
19	Shperical 6 $=\sqrt{\sum_i^6 x_i^2}$	$[-1,1]^6$	0.0	1
20	Shperical 7 $=\sqrt{\sum_i^7 x_i^2}$	$[-1,1]^7$	0.0	1
21	Shperical 8 $=\sqrt{\sum_i^8 x_i^2}$	$[-1,1]^8$	0.0	1
22	Shperical 9 $=\sqrt{\sum_i^9 x_i^2}$	$[-1,1]^9$	0.0	1

References

[1] Duckbury, P.G., *Parallel Array Processing.* Chichester: Ellis Horward, 1986.

[2] García, I., Ortigosa, P.M., Casado, L.G., Herman, G.T. and Matej, S., "A parallel implementation of the controlled random search algorithm to optimize an algorithm for reconstruction from projections," in *IIIrd Workshop on Global Optimization*, (Szeged, Hungary), pp. 28–32, December 1995.

[3] García, I. and Herman, G.T., "Global optimization by parallel constrained biased random search," in *State of Art in Global Optimization: Computational Methods and Applications* (C.A. Floudas and P.M. Pardalos, ed.), Kluwer Inc, pp.433-455, 1996.

[4] García, I., Ortigosa, P.M., Casado, L.G., Herman, G.T. and Matej, S., "Multidimensional Optimization in Image Reconstruction from Projections," in *Developments in Global Optimization*, (L.M. Bomze, T. Csendes, R. Horst and P.M. Pardalos eds,), Kluwer Inc, pp. 289–300, 1997.

[5] Hendrix, E.M.T., *Global Optimization at Work*, PhD. Dissertation, Wageningen Agricultural University, 1998.

[6] Horst, R. and Pardalos, P.M. eds., *Handbook of Global Optimization*, Dordrecht: Kluwer, 1995.

[7] McKeown, J.J., "Aspects of parallel computations in numerical optimization," in *Numerical Techniques for Stochastic Systems* (F. Arcetti and M. Cugiani, eds.), pp. 297–327, 1980.

[8] Nelder, J.A. and Mead, R., "A simplex method for function minimization," in *The Computer Journal*, pp. 308–313, 1965.

[9] Price, W.L., "A controlled random search procedure for global optimization," in *Towards Global Optimization 2* (L.C.W Dixon and G.P. Szegö, eds.), pp. 71–84, Amsterdam: North Holland, 1978.

[10] Price, W.L., "A controlled random search procedure for global optimization," in *The Computer Journal*, no. 20, pp. 367-370, 1979.

[11] Price, W.L., "Global optimization algorithms by controlled random search," *Journal of Optimization Theory and Applications*, no. 40, pp. 333–348, 1983.

[12] Price, W.L., "Global optimization algorithms for a CAD workstation," *Journal of Optimization Theory and Applications*, no. 55, pp. 133–146, 1987.

[13] Sokal, R.R. and Rohlf, F.J., *Biometry.* New York: W. H. Freeman and company, 1981.

[14] Sutti, C., "Local and global optimization by parallel algorithms for MIMD systems," *Annals of Operating Research*, no. 1, pp. 151–164, 1984.

[15] Törn, A. and Zilinskas, A., *Global Optimization. Lecture Notes in Computer Science 350.* Springer-Verlag, Berlin, 1989.

[16] Woodhams, F.W.D. and Price, W.L., "Optimizing accelerator for CAD workstation," *IEE Proceedings Part E*, vol. 135, no. 4, pp. 214–221, 1988.

CONTENTS

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Csirik János
A kézirat a nyomdába érkezett: 1999. október
Terjedelem: 13,7 (B/5) ív